HOLGER PROTHMANN

Organic Traffic Control

KIT Scientific Publishing

Holger Prothmann

Organic Traffic Control

Organic Traffic Control

by
Holger Prothmann

KIT Scientific Publishing

Dissertation, Karlsruher Institut für Technologie
Fakultät für Wirtschaftswissenschaften
Tag der mündlichen Prüfung: 15. Juli 2011
Referenten: Prof. Dr. Hartmut Schmeck, Prof. Dr.-Ing. Peter Vortisch

Impressum

Karlsruher Institut für Technologie (KIT)
KIT Scientific Publishing
Straße am Forum 2
D-76131 Karlsruhe
www.ksp.kit.edu

KIT – Universität des Landes Baden-Württemberg und nationales
Forschungszentrum in der Helmholtz-Gemeinschaft

KIT Scientific Publishing 2011
Print on Demand

ISBN 978-3-86644-725-7

Organic Traffic Control

Zur Erlangung des akademischen Grades eines
Doktors der Wirtschaftswissenschaften

(Dr. rer. pol.)

von der Fakultät für Wirtschaftswissenschaften
des Karlsruher Instituts für Technologie (KIT)

genehmigte

DISSERTATION

von

Dipl.-Inform. Holger Prothmann

Tag der mündlichen Prüfung:	15. Juli 2011
Referent:	Prof. Dr. Hartmut Schmeck
Korreferent:	Prof. Dr.-Ing. Peter Vortisch

2011 Karlsruhe

Acknowledgements

This thesis is the outcome of several years of research under the supervision of my doctoral adviser Prof. Dr. Hartmut Schmeck. During these years, I have greatly benefited from his extensive experience, his constructive remarks, and his fondness for detail. I am grateful for his support and his trust in the project's success.

I would like to thank Prof. Dr.-Ing. Peter Vortisch for accepting the request to serve as second reviewer for this thesis. Furthermore, my thanks go to Prof. Dr. Stefan Nickel and Prof. Dr. Kay Mitusch who completed the examination committee.

I am grateful to all colleagues with whom I have worked during the last years. At the Institute AIFB, the current and former members of the research group "Efficient Algorithms" provided a pleasant and productive working atmosphere. I would like to express my special thanks to my office mates Dr. Matthias Bonn and Christian Hirsch, as well as to Nugroho Fredivianus, Dr. Andreas Kamper, Lukas König, Lei Liu, Dr.-Ing. Sanaz Mostaghim, Dr. Urban Richter, and Dr. Pradyumn Shukla for productive, interesting, and encouraging discussions. Urban deserves a very special thank you for carefully proofreading the manuscript.

This work has also largely benefited from discussions with external project partners. I would like to express my gratitude to Fabian Rochner

and Sven Tomforde at Leibniz Universität Hannover as various parts of this thesis are based on an intensive collaboration with both of them. Prof. Dr. Jürgen Branke, Prof. Dr. Jörg Hähner, and Prof. Dr.-Ing. Christian Müller-Schloer made valuable contributions to this work by sharing their expertise. Moreover, I would like to thank Emre Çakar and Dr.-Ing. Moez Mnif for making the regular project meetings a pleasant experience.

Data used in this thesis has been provided by *Tiefbauamt* (Karlsruhe), *Schmeck Ingenieurgesellschaft mbH* (Hamburg), and *Landesbetrieb Straßen, Brücken und Gewässer* (Hamburg). I would like to thank these institutions for their support.

Last but not least, I am grateful to my family and all friends who supported me throughout the years.

Karlsruhe, August 2011 *Holger Prothmann*

Contents

CHAPTER 1

Introduction

While in 2008, for the first time, more than half of the world's population has lived in urban areas, the United Nations predict a continuing urbanisation during the coming years [199]. In consequence, the provision of an appropriate infrastructure – for water, power, and transportation – is becoming a challenge for many cities.

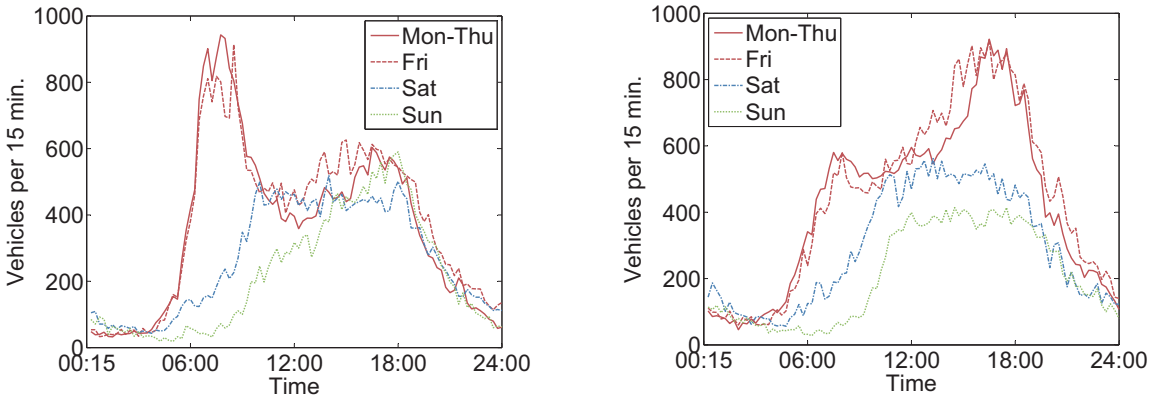
Efficient road networks are one of the infrastructural key factors as they facilitate the transportation of people and goods. While the demand for traffic and transportation is increasing, the road networks often cannot be extended at the same rate. In consequence, congestion is a widespread problem that leads to a negative environmental and economic impact [172]. The environment suffers from vehicular emissions and noise, while traffic jams waste people's time and result in considerable financial losses for the economy.

1.1 Motivation

As urban road networks are characterised by their numerous signalised intersections, optimised signal plans and an improved coordination can help to use the existing road network efficiently, thereby reducing the negative impact of traffic. Unfortunately, even the optimisation of a single signalised intersection is a task that usually cannot be solved analytically. Choosing adequate green times for an intersection's traffic movements requires a system of differential equations that in general has no closed form solution. In order to obtain a closed form solution, the underlying mathematical model needs to be simplified to a degree where it is no longer generally applicable [204].

An additional difficulty results from the fact that traffic demands in urban road networks are dynamically changing such that the signalisation needs to be continuously adapted. Figure 1.1 illustrates the changes in demand for the Südtangente, an arterial road at Karlsruhe, Germany, during different weekdays. In the figure, different weekdays can be clearly distinguished by their traffic demands. On workdays, the demand is characterised by pronounced peaks in the morning (eastward direction) and afternoon (westward direction) due to commuters travelling to and from the city centre, respectively. On Fridays, the evening peak starts earlier than on other workdays due to shorter working hours in many offices. Compared to workdays, the traffic demand on a Saturday rises later and shows no pronounced peaks. The number of vehicles is smaller than on a workday with a majority of travellers being on the road during the afternoon. The traffic demand on Sundays (or other holidays) is similar to that of a Saturday, with a further reduced number of travellers on the road. A weak peak can be observed in the late afternoon towards the city centre. The traffic demand visualised in Figure 1.1 is quite typical for arterial roads in many cities.

Figure 1.2 depicts the situation at an intersection in more detail. For a workday's morning and afternoon peak hour, the traffic flows for each turning are indicated by arrows of different widths. The arrow width is proportional to the traffic flow for the intersection's turnings. By comparing the figures for the morning and afternoon peak, it can be observed that the traffic flows for several turnings significantly change during the day.



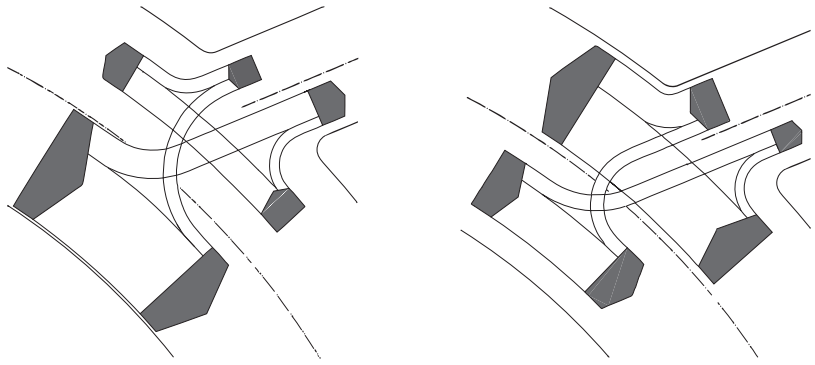
(a) Eastward direction

(b) Westward direction

Figure 1.1: Traffic demand of the Südtangente at Karlsruhe, Germany (Data provided by Tiefbauamt Karlsruhe)

Changes in demand need to be accompanied by changes in the intersection's signalisation. When changes occur on a regular basis, they can in principle be handled by a time-dependent switching of signal plans. However, setting up sufficiently detailed schedules requires a significant design effort and the resulting plans are inflexible and cannot cope with irregularities: During public events like soccer matches, concerts, or trade fairs, the traffic demand does not resemble the intra-day demand of any regular weekday. An example is given in Figure 1.3 that shows the traffic demand of the Südtangente on two subsequent Sundays. While June 20, 2010 has been a regular Sunday, Germany played England in the round of sixteen in the FIFA World Cup in the afternoon of June 27, 2010. Although the game took place in Bloemfontein, South Africa, it affected the traffic in Germany. During the game, traffic counts at the Südtangente were drastically reduced, while a slight increase can be observed for the rest of the day due to pre- or postponed trips.

Other events that result in irregular traffic demands include holidays that fall on a workday, demonstrations or strikes that are often announced on short notice, or blocked roads due to incidents or weather conditions. Events like these cannot be handled by a time-dependent switching of signal plans, since they are difficult or even impossible to foresee at design time. Therefore, it is necessary to shift the signal plan optimisation from



(a) Morning peak (b) Afternoon peak

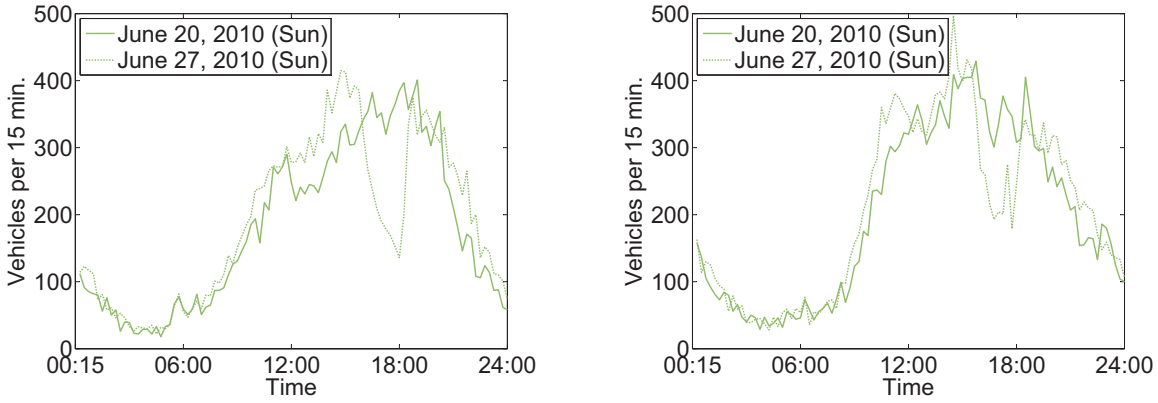
Figure 1.2: Traffic demands of an intersection at Hamburg, Germany (Data provided by Schmeck Ingenieurgesellschaft mbH)

design time to the run-time of the signal system. This calls for learning intersections that can autonomously reconfigure their signalisation online.

As several intersections can be located in close vicinity within an urban road network, their coordination is another important aspect. Figure 1.4 illustrates the effect of coordination by showing a time-distance diagram for an arterial road of three intersections. Each line in the diagram depicts the trajectory of a vehicle travelling along the arterial in northern direction. Stops appear as horizontal lines in the figure.

Figure 1.4a shows that vehicles can be stopped repeatedly while travelling along the arterial in case that the intersections are not properly coordinated. By coordinating their signals, the number of stops can be significantly reduced after the vehicles have passed the first intersection where they arrive randomly (see Figure 1.4b).

Reducing the number of stops in a traffic network can have beneficial effects on travel times, fuel consumption, and pollution emission. Unfortunately, signal coordination is a complex problem that involves several restrictions. In the general case, it is, e. g., not even possible to coordinate an arterial road along both its directions [171]. In addition to this complexity, dynamically changing traffic demands require changes in the coordination. This calls for intersections that can self-organise to achieve a traffic-responsive coordination within the road network.



(a) Eastward direction

(b) Westward direction

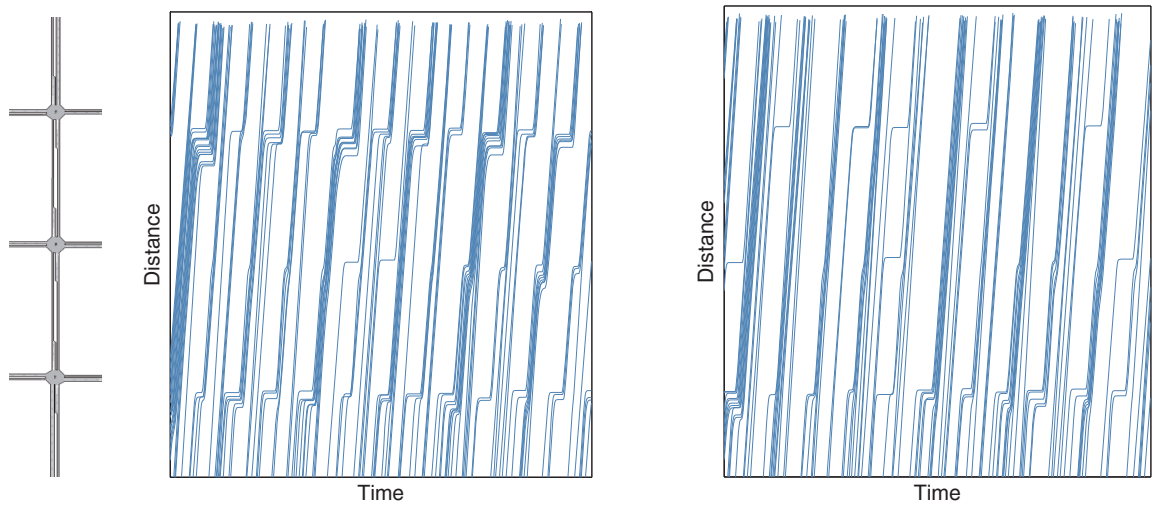
Figure 1.3: Traffic demand of the Südtangente at Karlsruhe, Germany, during the 2010 FIFA World Cup in South Africa (Data provided by Tiefbauamt Karlsruhe)

1.2 Objectives and approach

A research field that anticipates adaptive, self-organising technical systems is *Organic Computing* [137, 139, 168, 201]. The anticipated technical systems should be able to adjust autonomously to changes in their environment (like changing traffic demands) while being robust (with respect to disturbances or failures) and flexible (with respect to externally provided goals). Furthermore, the systems should exhibit learning capabilities and provide *self-x features* like self-configuration or self-optimisation. Due to these properties that are inspired by living organisms, such systems are called *organic*.

A powerful design framework for organic systems is the *observer/controller architecture* [23, 136, 157, 159]. As depicted in Figure 1.5, the architecture extends a technical *System under Observation and Control* (SuOC) with an observer/controller loop that continuously monitors the underlying SuOC and reconfigures it when this becomes necessary (e. g., due a changing system environment).

The reconfiguration relies on a machine learning mechanism that enables the observer/controller to handle environmental conditions that were unforeseen at design time, thereby fulfilling an important require-



(a) Uncoordinated intersections (b) Coordinated intersections

Figure 1.4: Time-distance diagram for an arterial road

ment for many adaptive systems. However, learning involves errors that should not negatively affect the functionality or performance of the SuOC. Many machine learning techniques rely on a preceding training phase to learn at design time or they test alternative configurations directly in the SuOC (thereby running the risk of reduced performance or system failure, which in some applications might be acceptable). Since both approaches are not satisfactory for an organic system, the observer/controller introduces a *two-levelled learning* that combines an on-line learning and an off-line optimisation level.

Off-line optimisation enables the observer/controller to find appropriate configurations for the SuOC based on a model of the system (e. g., a traffic simulation). Using the model, different configurations of the SuOC can be safely tested without affecting the productive system. Good configurations found by model-based optimisation become available to the on-line learning mechanism that memorises optimised configurations and activates them when necessary. The performance of activated configurations is tracked by the learning mechanism, such that their applicability can be updated based on feedback gathered in the SuOC.

An organic system nicely fulfills the requirements of an adaptive learning intersection. The signalised intersection becomes the SuOC that is

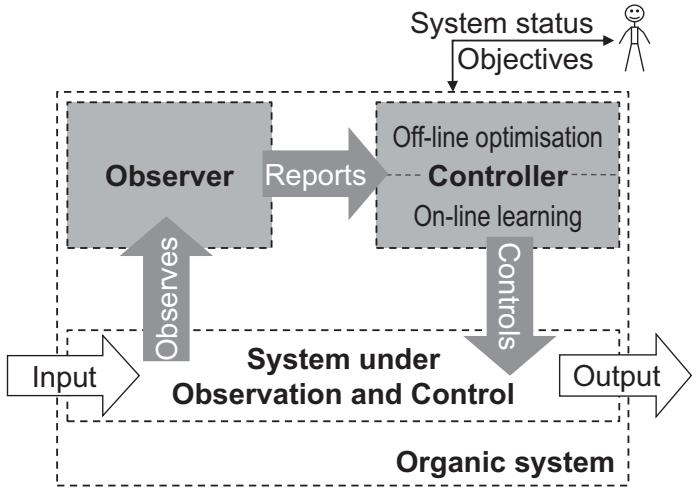


Figure 1.5: Simplified observer/controller architecture

equipped with an observer/controller monitoring the local traffic demands and reconfiguring the signalisation accordingly. The two-levelled learning employed by the observer/controller facilitates to shift the configuration of signal plans from design time to run-time, thereby allowing the intersection to handle even irregular traffic demands. Therefore, the first two main objectives of this thesis are:

1. Implementing the observer/controller architecture for traffic signal control to obtain a signalised intersection with organic properties

2. Investigating two-levelled learning as an approach for machine learning in safety- and performance-critical environments

The third main objective results from the fact that signalised intersections in urban networks are located in the vicinity of each other. This gives special importance to a coordinated operation of signals at neighbouring intersections in order to achieve a reduction of stops in the network which, as a result, also reduces travel times, fuel consumption, and pollution emissions. Most existing network control systems rely on predefined coordination schemes or achieve coordination through a centralised traffic control centre. As the centralised coordination of signalised intersections is complex, monetarily costly, and potentially susceptible to failures, there is a need for decentralised or hierarchical solutions that

shift (parts of) the decision process to the intersections [74, 134]. As Organic Computing envisions self-organising systems that consist of locally interacting components which obtain an emerging global behaviour without centralised control, thereby resulting in a scalable architecture that does not suffer from bottlenecks, the third main objective of this thesis is:

3. Developing a self-organising coordination mechanism for intersections in urban road networks

The mentioned objectives directly relate to the major contributions of this thesis which affect the areas of Organic Computing, machine learning, and traffic engineering.

1.3 Major contributions

The first contribution is an organic signal controller for intersections that is based on the generic observer/controller architecture. Due to its learning and optimisation capabilities, the controller autonomously adapts its signalisation to dynamically changing traffic demands which results in an improved performance with respect to a user-defined objective (like the average vehicular delay). By shifting the configuration of signal plans from design time to the run-time of the signal system, organic intersections can handle traffic demands that have not been anticipated by the designing engineer, thereby reducing the effort required for setup and maintenance. During learning, the observer/controller autonomously creates a set of human-readable rules that map traffic demands to signal plans, such that an understandable, transparent system behaviour is guaranteed. Moreover, the developed observer/controller is widely applicable since the only prerequisite is a reconfigurable signal controller equipped with traffic detectors.

While the first major contribution is mainly related to Organic Computing and traffic engineering, the second contribution affects the field of machine learning. The two-levelled learning mechanism that combines on-line reinforcement learning and off-line optimisation addresses the *exploration-exploitation dilemma* [186] frequently encountered in reinforcement learning. The dilemma arises after a reinforcement learning

mechanism accumulated some knowledge about its environment and has to decide on either exploiting this knowledge or exploring new actions. By opting to explore, the learner can potentially improve its long term performance since it might find a higher rewarded action. However, the learner runs the risk of drastically reducing its short term performance since exploration is likely to result in bad actions. Therefore, opting to exploit the previously learnt knowledge might be the better alternative.

Two-levelled learning addresses the exploration-exploitation dilemma by separating exploitation and exploration. On the exploration level, an optimisation algorithm (e. g., an Evolutionary Algorithm [62]) safely and quickly explores potential actions, relying on a (simulation) model of the environment in the process. On the exploitation level, a reinforcement learning mechanism (e. g., a Learning Classifier System [37]) memorises the exploration results, exploits the learnt actions, and updates their performance evaluation based on a reinforcement received from the environment. In consequence, two-levelled learning avoids costly explorations in the environment by the clever use of an environmental model while being applicable to a wide area of machine learning problems.

The third major contribution (that relates to the fields of Organic Computing and traffic engineering) is a self-organising coordination mechanism that allows for the traffic-responsive coordination of signalised intersections in urban road networks. The mechanism dynamically creates *progressive signal systems* (or *green waves*) in response to the current traffic flows and is available in two architectural variants.

A completely decentralised variant establishes progressive signal systems by local communication among neighbouring intersections. Thus, the coordination mechanism is scalable and avoids the bottleneck of a traffic control centre. Coordinated signal plans are determined locally according to the current traffic demand using the organic intersections' observer/controller.

In the hierarchical variant of the mechanism, the signalised intersections remain autonomous entities that collaborate locally to establish progressive signal systems, but a regional component supports the coordination process by resolving conflicts among competing traffic streams. Both architectures achieve a reduction of stops which is also beneficial with respect to fuel consumption and pollution emission.

1.4 Thesis overview

At a glance, this thesis consists of nine chapters, starting with this introduction.

Chapter 2 introduces the basic concepts of fixed-time and traffic-actuated signal control and defines related terms that are used throughout this thesis. The coordination of signalised intersections is discussed and selected network control systems are reviewed. Since these systems typically rely on simulation models, the foundations of micro- and macroscopic traffic simulation are briefly presented.

Chapter 3 discusses optimisation approaches for signalised traffic networks and presents learning and self-organising control mechanisms. With respect to signal plan optimisation, the chapter is based on a previous publication [147] and focuses on Evolutionary Algorithms as nature-inspired heuristics. Their working principle is introduced and selected applications to off-line optimisation (at design time) and on-line optimisation (at run-time) are reviewed. Learning Classifier Systems are introduced as an example for a machine learning technique that can be applied to implement learning intersections. Finally, selected self-organising control mechanisms (that coordinate a network's traffic signals based on local interactions) are presented.

While the previous chapters present the state of the art in traffic signal control and discuss learning and optimisation approaches, Chapter 4 is dedicated to Organic Computing, a research field that investigates how to design adaptive and self-organising technical systems. Being partly based on [23], the chapter introduces the observer/controller architecture that provides a generic design framework for organic systems and reviews related frameworks.

In Chapter 5, the observer/controller architecture is implemented to obtain an adaptive learning intersection controller. Being partly based on [145, 146, 161], the chapter presents the monitoring of local traffic demands by the observer and discusses the controller's two-levelled learning mechanism that uses a Learning Classifier System for signal plan selection and an Evolutionary Algorithm for simulation-based signal plan optimisation.

In Chapter 6, the performance of organic intersections is experimentally evaluated. The chapter identifies crucial components in the ob-

server/controller and presents a sensitivity study to determine their configuration. Using the findings of the study, the organic intersection is experimentally evaluated based on traffic simulations of several real-world intersections.

The self-organised coordination of intersections is in the focus of Chapter 7. The chapter is partly based on [145,192] and presents a decentralised coordination mechanism that allows for the traffic-adaptive creation of progressive signal systems in urban roads networks. The mechanism relies on local detection data and communication among neighbouring intersections. It is investigated in a simulation study to assess its potential savings with respect to travel time, stops, and, in consequence, fuel consumption and pollution emission.

In Chapter 8 – that is partly based on [191] – the coordination mechanism is extended by a Regional Manager. Relying on a network-wide traffic model, the manager supports the signal coordination by solving conflicts of interest among the network's traffic streams. As the signalised intersections remain autonomous entities that collaborate locally to establish progressive signal systems, the Regional Manager transforms the formerly decentralised mechanism into a hierarchical architecture. Both system variants are experimentally compared to investigate the possibilities and limitations of decentralised traffic control.

To conclude the thesis, Chapter 9 summarises its main contributions and outlines promising directions for future work.

1.5 How this thesis was written

This thesis is the outcome of several years of research with financial support by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) within the Priority Programme 1183 OC. Several chapters of the thesis are partly based on papers that have been published with different colleagues from the research group of my doctoral adviser, Prof. Dr. Hartmut Schmeck (Karlsruhe Institute of Technology), and from the groups of Prof. Dr.-Ing. Christian Müller-Schloer, Prof. Dr. Jörg Hähner (both from Leibniz Universität Hannover), and Prof. Dr. Jürgen Branke (University of Warwick).

CHAPTER 2

Traffic control and simulation

This chapter discusses foundations of traffic control and simulation. Section 2.1 introduces basic signalisation concepts and defines important terms that are used throughout this thesis. Section 2.2 focuses on fixed-time controls and their configuration. More sophisticated traffic-actuated controls that adapt their signalisation based on predefined conditions are discussed in Section 2.3. Adaptive network control systems that continuously evaluate and optimise a network's traffic signals are in the focus of Section 2.4. These systems typically rely on traffic models which are discussed in Section 2.5, where foundations of micro- and macroscopic simulation models are presented. Finally, the chapter closes with a short summary and a contextual classification in Section 2.6.

2.1 Foundations and terms

Traffic signals are installed at an intersection in order to reduce or eliminate conflicts among traffic movements. A *traffic movement* (or

turning movement) combines road users of the same class (e. g., motorised vehicles, cyclists, or pedestrians) that reach the intersection on the same approach and leave it using the same exit. Conflicts occur when different movements intersect on their way.

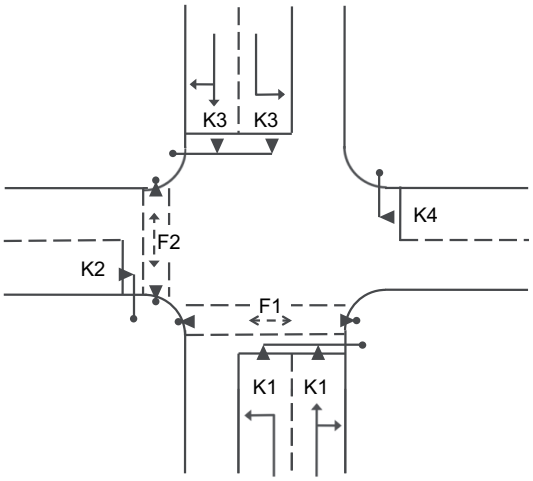
Traffic lights allocate green time to the various movements. Signals that cannot be switched separately due to their electric wiring – and therefore show the same signalisation at any point in time – form a *signal group*. Several signal groups that commonly show a green signal for a certain period of time form a *phase*. Phases are the basic states of the signalisation. When phases change, *phase transitions* control the switching operations of the involved signal groups to ensure sufficient *clearing times* between ending and starting traffic movements. By assigning each phase a duration and including the relevant phase transitions, a *signal plan* is obtained. The signal plan visualises the resulting *green times* for the intersection's signal groups as well as the corresponding *red times*.

Figure 2.1 illustrates the basic definitions. A simple example intersection with six signal groups is shown in Figure 2.1a. The indicated signal groups K1 to K4 control the vehicular traffic movements at the intersection, while the signal groups F1 and F2 provide the signalisation for pedestrian movements. For exemplification, the signal groups have been combined into two phases, as depicted in Figure 2.1b. A corresponding signal plan is shown in Figure 2.1c, where phases and phase transitions are highlighted. Additionally, green, red, and clearing times for exemplary signal groups are indicated in the figure. Amber and red with amber periods that separate the red and green periods are omitted for simplification.

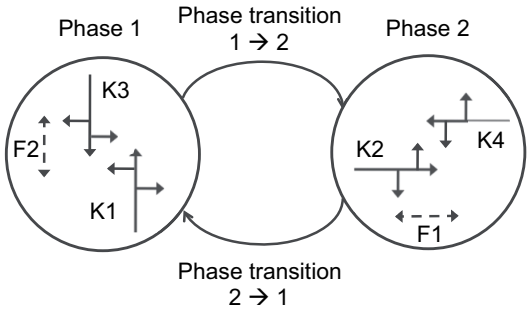
In the following sections, fixed-time and traffic-actuated controls are briefly introduced. Comprehensive instructions on their setup, operation, and maintenance that are relevant for Germany are available in the *Richtlinien für Lichtsignalanlagen* (RiLSA, [70]).

2.2 Fixed-time control

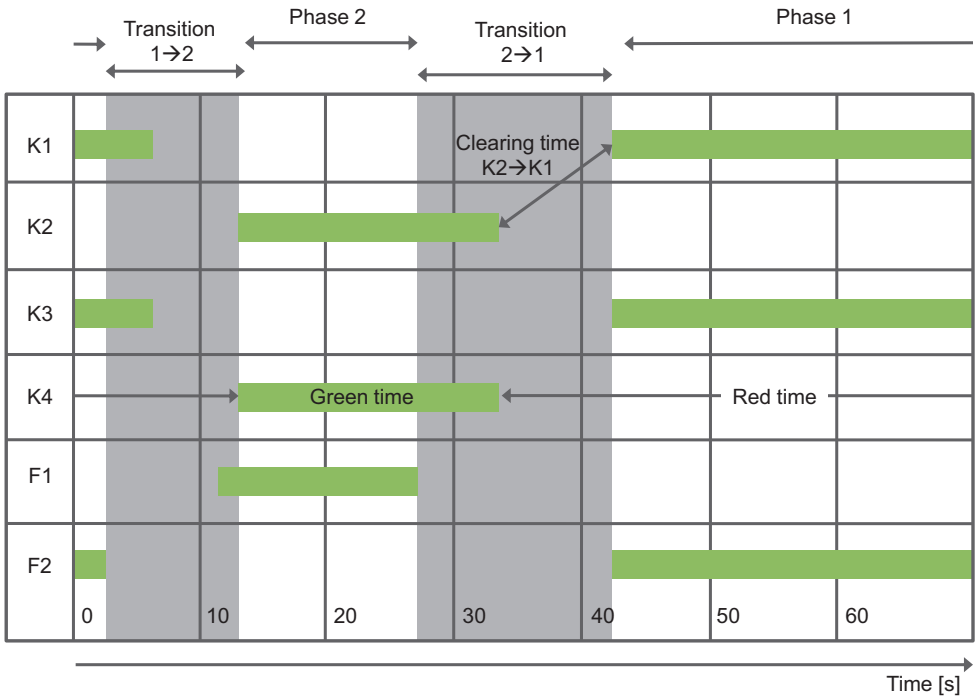
Fixed-time controls operate based on predefined signal plans, no traffic-responsive change of the signalisation takes place. Due to their simplicity, fixed-time controls are still in use today. Section 2.2.1 introduces their



(a) Intersection with signal groups

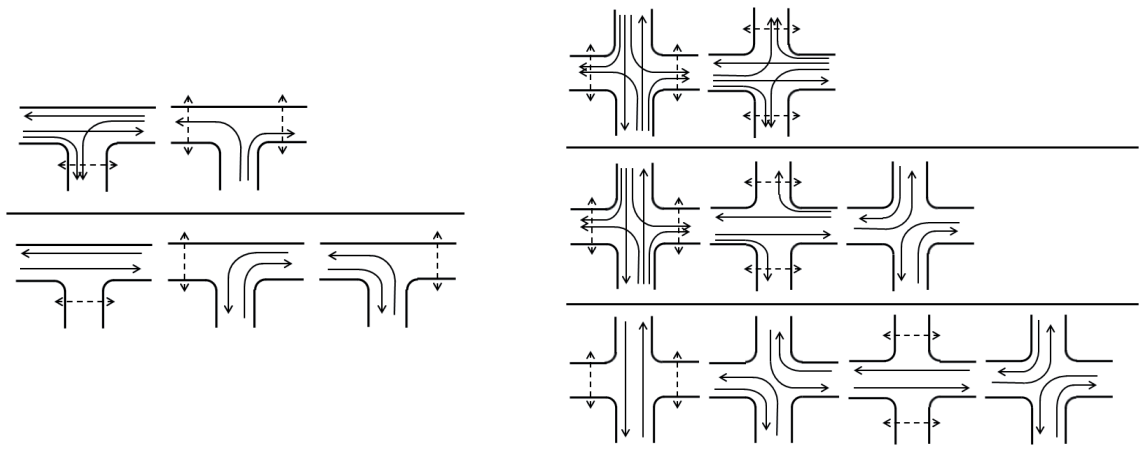


(b) Phase sequence with phases and transitions



(c) Signal plan

Figure 2.1: Intersection with signal groups, phase sequence, and signal plan (based on [127])



(a) Three-armed intersection (b) Four-armed intersection

Figure 2.2: Phase systems

working principle, while Section 2.2.2 discusses the derivation of approximately delay-optimal signal timings.

2.2.1 Working principle

Fixed-time control is the oldest and simplest way to operate signalised intersections. A fixed sequence of phases with predefined durations is periodically repeated such that each signal group obtains the right of way at least once within the cycle. At fixed-time controlled intersections, the signalisation is not influenced by the current traffic flows, but is completely predetermined by the following basic signal settings:

Phase sequence Starting with the intersection's signal groups, phases need to be defined and arranged in a fixed sequence. Basic phase systems with two, three, or four phases are depicted in Figure 2.2 for three- and four-armed intersections. The phase systems mainly differ in the handling of left-turning vehicle movements that are either signalised separately or in combination with the opposing traffic.

Once defined, the phases can be arranged in different sequences that lead to different phase transitions and clearing times. As a result, the phase sequence influences the obtainable green times.

Cycle time The cycle time corresponds to the time available for one iteration through the signal plan. The cycle should be short to avoid unnecessary delays, but not too short since otherwise the clearing times make up a too large fraction of the cycle and thereby reduce the effectively available green times. Details on the determination of an approximately delay-optimal cycle time are given in Section 2.2.2.

Phase durations The phase durations (or *splits* when defined relative to the cycle time) determine the amount of time available for each phase. Section 2.2.2 presents details on the determination of approximately delay-optimal phase durations.

The basic signal settings mentioned above have to be specified by a traffic engineer at design time. The specifications are based on past traffic demands (obtained, e. g., from a traffic census) and incorporate the engineer's expert knowledge. To handle intra-day changes in traffic, a day-time dependent switching of signal plans is commonly used. Typically, separate signal plans are derived for day and night periods and for peak hours occurring during the morning and evening.

Long term traffic developments, however, cannot be handled in this way. Since fixed-time controls are known to suffer from an *ageing effect* that reduces their quality over time [17], regular signal plan reviews by a traffic engineer are required to maintain a constant signalisation quality.

An advantage of fixed-time control is that neighbouring intersections can be easily coordinated such that vehicles can pass several nodes without encountering a red signal. Coordination is achieved by operating the intersections with different *offsets* that are defined as the difference between a reference time and the start time of the phase serving the coordinated traffic movement. By adjusting the offset difference among adjacent intersections to equal the travel time between those intersections, a coordination can be established. A prerequisite, however, is a common cycle time for the coordinated intersections to ensure that the phases remain synchronised over time. The cycle time is generally determined by the most heavily used intersection, but if nodes with significantly lower traffic demands are part of the coordination, they can operate at an integer fraction of the cycle. In these special cases, a full coordination is established in every second or third cycle, only.

Figure 2.3 shows the signalisation of three coordinated intersections with a common cycle time in a time-space diagram. The horizontal and vertical axes represent distance and time, respectively. The illustrated signalisation (obtained by offset adjustments) allows for an unimpeded passage of vehicles (travelling at a constant speed) within the highlighted green band.

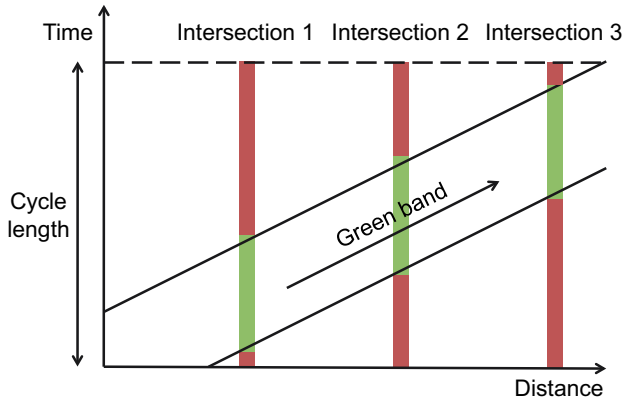


Figure 2.3: Coordination of traffic signals

More complex coordination schemes are possible (e. g., along both directions of an arterial), but their applicability is strictly limited by the distance between the coordinated intersections. In traffic networks, the situation is even more complex since a coordination can be disadvantageous for road users travelling along uncoordinated paths.

Regardless whether intersections are coordinated, a good configuration of the basic signal settings is essential for their efficient operation. Therefore, the following section focuses on the determination of approximately delay-optimal cycle times and phase splits.

2.2.2 Configuration

Several authors proposed methods to obtain signal timings that are – under certain assumptions – optimal with respect to the average vehicular delay at a signalised intersection (see [58,100] for an overview). A method that is widely used due to its relative simplicity has been proposed by Webster [208]. In the following, Webster's method is presented, starting

with an approximation formula for the average vehicular delay at a signalised intersection.

Approximating the average delay of a vehicle

The delay of a vehicle at an intersection compares the time the vehicle needs to pass the stop line of a signalised node to the time of an undisturbed passage with maximum allowed speed. More formally, the vehicle's delay combines the delay times resulting from slowing down, waiting at a red light, and speeding up again after the stop.

Figure 2.4 visualises the delay by depicting the corresponding vehicle trajectories in a time-space diagram: The dashed trajectory represents a vehicle that passes the intersection's stop line with no stop, while the solid trajectory belongs to a stopped vehicle that slows down, waits, and speeds up again. The resulting delay is depicted in the figure.

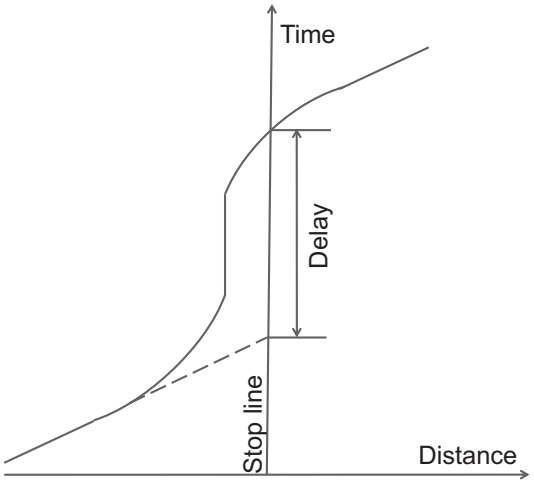


Figure 2.4: Delay of a vehicle at a signalised intersection (based on [171])

The vehicular delay at a signalised intersection can be calculated by combining the average delays of all vehicular traffic movements. Making the simplifying assumptions that

- the vehicle arrivals are uniformly distributed and that
- all vehicles can leave the queue during the subsequent green period,

these delays can be mathematically derived. Assuming that M corresponds to a turning's current traffic flow (in veh/h) and that S denotes its *saturation flow* – which is the maximal flow (in veh/h) that could theoretically occur assuming constant green – the average delay t'_d can be computed as

$$t'_d = \frac{t_C \cdot (1-f)^2}{2 \cdot (1 - M/S)}, \tag{2.1}$$

where t_C denotes the intersection's cycle time and f corresponds to the fraction of the turning movement's effective green time t_g in comparison to the cycle (i. e., $f = t_g/t_C$). The *effective green time* of a movement is the sum of its green and amber time less the time lost due to starting delays and reduced discharge rates during the amber period.

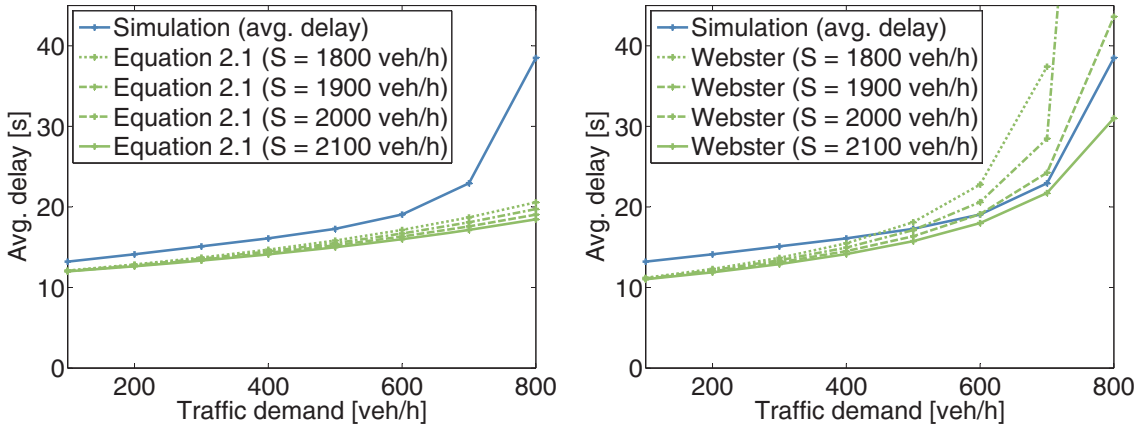
Although Equation 2.1 is completely based on theoretical considerations (see [171] for a detailed derivation), the equation relies on strong assumptions. At real-world intersections, vehicle arrivals are typically not uniformly distributed. Therefore, uncleared queues commonly exist at the beginning of red periods and lead to increased delays. Comparing the average vehicular delay obtained using a microscopic traffic simulator to the delay calculated with Equation 2.1, the underestimation becomes obvious especially for higher flows (see Figure 2.5a).

To obtain a more realistic delay estimation in case of Poisson distributed vehicle arrivals, Webster conducted simulation experiments and extended his theoretical considerations for uniformly distributed arrivals (see Equation 2.1) with an experimentally derived term. Here, the resulting formula is presented in a slightly simplified version that provides good estimates. According to Webster, the average delay t_d of a vehicle in case of Poisson distributed arrivals can be calculated as

$$t_d = 0.9 \cdot \left[t'_d + \frac{1800 \cdot g^2}{M \cdot (1-g)} \right], \tag{2.2}$$

where g corresponds to the degree of saturation of a turning (i. e., $g = M/(f \cdot S)$).

Figure 2.5b compares the average vehicular delay calculated using Webster's formula to data obtained from a microscopic traffic simulator. It can be observed that the formula provides a good approximation of the simulation results provided that the chosen saturation flow estimate



(a) Equation 2.1 (b) Equation 2.2 (Webster)

Figure 2.5: Comparison of average delays measured in a microscopic simulator or calculated using approximation formulas ($t_C = 70\,\mathrm{s}$, $t_g = 30\,\mathrm{s}$)

S reflects the real-world conditions well. When the chosen value of S is too small, the traffic flow M nearly reaches the capacity $C = f \cdot S$ of the signalised turning movement and therefore the degree of saturation $g = M/C$ becomes nearly 1. For these cases, it is known from the literature (and can be observed in the figure) that Webster's formula overestimates delays [171].

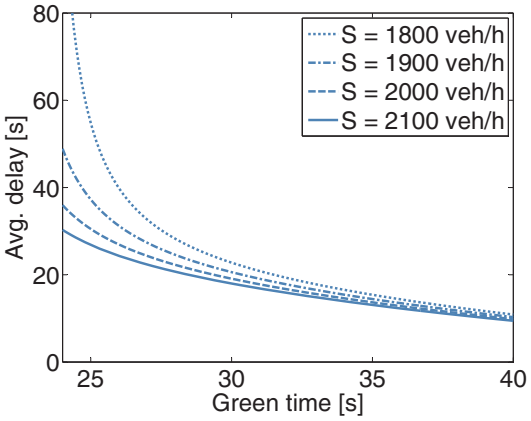
To estimate the average delay for a signalised intersection, the delays computed for all turning movements can be combined in a weighted sum. Assuming that M_i is the traffic flow for the i-th turning while $t_{d,i}$ specifies the turning's average delay computed according to Equation 2.2, the average delay t_D of the intersection can be calculated as

$$t_D = \frac{\sum_i (M_i \cdot t_{d,i})}{\sum_i M_i}. \tag{2.3}$$

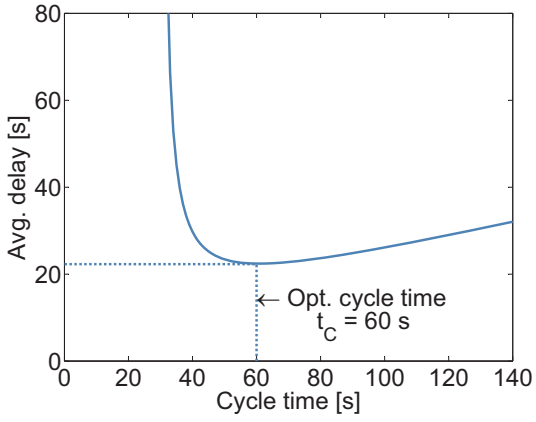
The average delay of an intersection calculated according the Equation 2.3 is a widely used measure for the quality of signal programmes. Therefore, the effects of signal timings on the resulting delay is briefly investigated in the following.

Effects of signal timings on the vehicular delay

Using the approximation formulas of Webster, the effect of varied green and cycle times on the vehicular delay can be clarified. Figure 2.6a illustrates the relationship of available green time and average delay for a signalised turning movement of a fixed-time controlled intersection. The figure was derived using Equation 2.2 for an assumed flow of 600 veh/h and and a cycle time of 70 s using varying saturation flows. As expected, increased green times lead to reduced average delays, while the reduction of available green time below a (saturation flow-dependent) threshold leads to a disproportionate increase of delays. It is therefore desirable to choose the green time of a turning movement as large as possible, ensuring that on average all arriving vehicles can be served within the green period. At an intersection, however, the available green time for a turning is limited by the time requirements of conflicting turning movements.



(a) Average delay for a turning depending on the green time ($t_C = 70\,\text{s}$, $M = 600\,\text{veh/h}$)



(b) Average delay for an intersection depending on the cycle time (Two-phased intersection, $M_1 = M_2 = 600\,\text{veh/h}$, $S_1 = S_2 = 1800\,\text{veh/h}$, $t_{g,1} = t_{g,2} = 0.5 \cdot t_C - 5\,\text{s}$)

Figure 2.6: Effect of green and cycle times on observed delays

Besides phase durations, the cycle time is an important parameter of fixed-time controlled intersections. Its effect on the average vehicular delay at an intersection is depicted in Figure 2.6b. The figure has been derived using Equation 2.3 for an intersection with two conflicting turning

movements with a flow of 600 veh/h and a saturation flow of 1800 veh/h each. The effective green times for the phases are assumed as half of the cycle time minus 5 s, i. e., $t_{g,1} = t_{g,2} = 0.5 \cdot t_C - 5$. It can be observed that for the optimal cycle time – which is 60 s in the example – small changes of approximately 10 s result in marginally increased delays, only. While a further increase of the cycle time leads to slowly increasing delays, a further decrease towards the minimal cycle time required to serve the arriving traffic results in a rapid rise of the resulting delay. This can be explained by the following considerations: For shorter cycle times, the lost times (which are independent of the cycle length) make up a larger fraction of the cycle which results in a reduced capacity of the intersection. Overly long cycle times, on the other hand, can lead to unused green times in one phase which unnecessarily increases the delay for other phases. Therefore, the cycle time of an intersection should be selected long enough to provide a sufficient capacity, but no overly long to avoid unnecessary delays.

In his work, Webster proposed an approach for the calculation of approximately delay-optimal signal-timings for fixed-time controlled intersections that is briefly presented in the following.

Optimal signal timings with respect to the vehicular delay

Based on his approximation formula for the average vehicular delay (see Equation 2.2), Webster derived equations to determine delay-optimal cycle times and phase durations for fixed-time controlled intersections. For the calculation of these timings, each phase is represented by a single turning movement that is served by the phase and exhibits the highest ratio of flow to saturation flow.

Cycle time According to Webster's considerations (see [208] for a detailed derivation), the optimal cycle time with respect to the resulting delays can be calculated as

$$t_C = \frac{1.5 \cdot t_L + 5}{1 - B}, \tag{2.4}$$

where t_L corresponds to the *total lost time per cycle* which sums up the average lost times t_l for all p phases and the all-red time t_R of

the cycle (i. e., $t_L = p \cdot t_l + t_R$). The *lost time per phase* (denoted by t_l) is the average amount of time lost due to starting delays and a reduced discharge rate during the amber period and is estimated to be 2 s by Webster. The *all-red time* t_R sums up all times during a cycle when all signals display red (or red with amber). Finally, B sums up the fractions of traffic flow M_i and saturation flow S_i for the representative turnings of the intersection's phases, i. e.,

$$B = \sum_{i=1}^{p} b_i = \sum_{i=1}^{p} \frac{M_i}{S_i}.$$

Phase durations To obtain minimal average delays, the (effective) phase durations $t_{g,i}$ should be in proportion to the corresponding ratios of flow to saturation flow, i. e., the relationship

$$t_{g,1} : t_{g,2} : \cdots : t_{g,p} = b_1 : b_2 : \cdots : b_p = \frac{M_1}{S_1} : \frac{M_2}{S_2} : \cdots : \frac{M_p}{S_p}.$$

should be fulfilled. This leads to effective phase durations of

$$t_{g,i} = \frac{b_i}{B} \cdot (t_C - t_L), \tag{2.5}$$

or green times (that consider lost times) of

$$t^*_{g,i} = t_{g,i} + t_l.$$

Equations 2.4 and 2.5 can be used to calculate approximately optimal cycle times and phase durations for fixed-time controlled intersections. However, Webster's method is subject to some restrictions:

- Webster's formulas work well whenever the traffic flow is light. For a high degree of saturation, however, singular events can heavily influence the traffic flow. In these cases, the applicability of Webster's method is limited.
- Webster's method cannot be directly applied to fixed-time traffic signals that serve the same signal group in several phases. Ohno and

Mine proposed a refinement of Webster's method that is not affected by these restrictions [143]. However, their suggested approximation algorithm is relatively complex and involves solving several linear programming problems.

- Webster's method is not applicable to traffic-actuated controls.

Despite these limitations, Webster's method is widely used to support the configuration of fixed-time controls. In this thesis, Webster's delay estimation (Equation 2.3) is combined with microscopic traffic simulations to provide fast and accurate fitness estimations for the evolutionary optimisation of signal timings (see Chapter 5).

2.3 Traffic-actuated control

In contrast to fixed-time operated intersections, traffic-actuated controls adapt their signalisation to the detected traffic situation. Adaptations are controlled by predefined temporal and logical conditions that represent the expert knowledge of the designing traffic engineer. Depending on the controller, traffic-actuated operations include some or all of the following modifications:

Green time adaptation The duration of signal phases can be varied within predefined boundaries. Depending on the number of waiting vehicles at the beginning of a phase or on gaps in the approaching traffic, phases can be extended or shortened, respectively. However, a green time adaptation does not influence the cycle time which is important when traffic-actuated controls are coordinated. The traffic-actuated controller guarantees that changes for one phase are compensated by other phases while all minimum and maximum phase durations are kept.

Green time adaptation is also used for public transport prioritisation. When a public transport vehicle is detected, the corresponding phase can be extended or conflicting phases can be shortened to allow for an undisturbed passage of the vehicle.

On-demand phases Some traffic-actuated controls support the on-demand inclusion of signal phases in their cycle. On-demand phases with no

waiting vehicles can be skipped, their green time is then subdivided among the remaining phases.

Phase sequence adaptation By changing the phase sequence, public transport vehicles can be prioritised. When an approaching public transport vehicle is detected, the controller brings forward the corresponding phase. Allowable phase transitions need to be predefined by a traffic engineer.

Cycle time adaptation Finally, the cycle time can be subject to modification if no coordinated operation is desired.

Temporal and logical conditions for the supported adaptations need to be defined in the planning phase of a traffic-actuated control based on the experience of a traffic engineer. During its operation, no quantitative evaluation and therefore no optimisation of control decisions takes place. As a consequence, traffic-actuated controls tend to loose their adaptivity and behave like fixed-time controls in heavy traffic [30], since phases are typically extended to their maximal duration in these conditions.

Traffic-actuated adaptation of the signalisation is usually limited to a single intersection, but a coordinated operation of several intersections is possible. The working principle is similar to the coordination of fixed-time controls (see Section 2.2): After fixing the cycle time of coordinated intersections, a coordination is achieved by adjusting the offsets for the coordinated phases. Traffic-actuated changes of the signalisation may affect non-coordinated phases only. The coordinated phases, however, must always be served at a fixed time and for a specified duration during each cycle in order to maintain the coordination.

2.4 Adaptive network control systems

Fixed-time and traffic-actuated controls are configured once at design time. During their operation, no further evaluation of control decisions takes place. In contrast, adaptive network control systems evaluate and optimise their control decisions on-line. Traffic flows and signal controls are usually modelled by the system and allow to determine the impact of control decisions on the traffic performance. Using these

performance measures, optimisation components search for the best possible signalisation for the current traffic demand.

This section discusses selected adaptive network control systems. Sections 2.4.1 and 2.4.2 present SCOOT and SCATS, two well-known centralised systems. The decentralised system OPAC is discussed in Section 2.4.3, while BALANCE and MOTION – two recently developed approaches – are presented in Section 2.4.4. Besides the systems discussed here, several other adaptive network control systems have been developed and are used throughout the world. More extensive overviews and reviews are available in the literature [75, 144, 180].

2.4.1 SCOOT

The *Split, Cycle, and Offset Optimisation Technique* (SCOOT, [160]) is one of the first adaptive network control systems that was successfully applied in the field. Since its development, it has been used in many installations worldwide and is still commercially available today. SCOOT centrally computes a single cycle time for all intersections in the network and adjusts phase durations (splits) and offsets for the individual nodes in order to reduce delays and stops.

The operation of SCOOT relies on a mesoscopic traffic model. The model combines data on advancing vehicles sensed by detectors at the upstream end of approaching links with observations from previous cycles to obtain *cyclic flow profiles*. In combination with information on expected speeds, saturation flows, and the intersection's signalisation, these profiles can be used to calculate queue lengths, delays, and stops for the network's intersections. Based on these performance measures (and in combination with the degree of saturation of links and intersections), splits, cycle time, and offsets can be adjusted:

Splits For split adjustment, SCOOT investigates at the end of each phase whether shortening or extending the phase reduces the degree of saturation for the most heavily saturated approach. Depending on the result of the check, the phase durations are adapted in small steps of a few seconds.

Cycle time Every few minutes, the cycle time is adjusted to achieve a degree of saturation of 90 % for the most heavily saturated intersec-

tion. As with split adaptations, the cycle time is changed in small steps of a few seconds.

Offset In contrast to cycle time and splits – which are adapted mainly based on the degree of saturation – the offset is adapted once per cycle using performance measurements derived from the cyclic flow profiles that have been computed for the incoming links. Again, the adaptation is performed in small steps.

By iteratively repeating the above steps, SCOOT adapts the signalisation to the current traffic flows in the network. The benefits obtainable by SCOOT installations are documented in several studies (e. g., [126]). However, SCOOT is also criticised for its stepwise change of control parameters that results in a relatively slow adaptation process [75].

2.4.2 SCATS

In the *Sydney Coordinated Adaptive Traffic System* (SCATS, [125, 175]), regional computers perform the strategic control of traffic-actuated intersections. Each regional system selects a common cycle length, splits, and offsets for the intersections within its subnetwork. Locally, the resulting signalisation can be modified by traffic-actuated operations. A centralised monitoring system is solely provided for the observation of the regional systems, their performance, and the equipment status.

In contrast to most adaptive network control systems, SCATS does not utilise a traffic model. The cycle time is calculated to maintain a degree of saturation between 80 % and 90 % for the most heavily saturated approach in the subnetwork whenever this is possible. Relying on a library of predefined controls, splits are adapted to maintain an equal degree of saturation on competing intersection approaches. Offsets are selected from predefined offset plans to suit the high flow movements in the subnetwork. All updates are performed on a cycle-to-cycle basis in order to optimise a performance criterion that can vary with the traffic demand. SCATS might, e. g., try to achieve

- minimum stops during periods of light demand,
- minimum delay with normal demand, and

- maximum throughput during periods of heavy demand.

SCATS is successfully used in many installations around the world with a geographical focus on Australia. Recently, the benefits of SCATS have been documented in a simulation study that considered an arterial of six intersections operated by coordinated fixed-time control [211].

2.4.3 OPAC

The *Optimisation Policies for Adaptive Control* (OPAC, [78]) have been developed to support the traffic-responsive control of single intersections. In contrast to SCOOT, SCATS, and most other adaptive network control systems, OPAC works decentralised in the sense that its control algorithm is applied independently at each intersection. For intersection control, OPAC relies on *dynamic programming* to compute a local signal switching policy that is optimal for a short time horizon in the future. Using a rolling horizon approach, the derived policies are continuously applied and recalculated in short intervals. As a result, intersections are operated acyclic and without an explicit coordination.

Although the signal switching plans derived by dynamic programming are in theory optimal with respect to the considered horizon, OPAC requires some adaptations in practice [78]:

- OPAC's policy calculation requires detailed vehicle arrival data for the entire horizon. Such data cannot usually be obtained from available detection systems, therefore a prognosis model is needed to substitute the missing data.

- The dynamic programming approach of OPAC requires an extensive computational effort that prevented its application in an on-line system. To reduce the computational requirements, a technique called *Optimal Sequential Constrained Search* (OSCO) replaced the original dynamic programming approach. Regarding OSCO, Gartner reports that it "provides results that are very close (within 10 %) to the genuine Dynamic Programming approach" [78].

Considering the mentioned adaptations, the basic steps in OPAC's operation can be summarised as follows:

1. Obtain the flow data from the intersection's detectors and assign this data to the head of the prognosis horizon. Estimate the flow data for the remaining horizon using the prognosis model.

2. Calculate an optimal switching policy for entire horizon using OSCO.

3. Implement the calculated switching policy for the head of the horizon only.

4. Shift the horizon and repeat Steps 1 to 4.

The literature reports that OPAC can offer remarkable benefits compared to traffic-actuated controllers [79]. OPAC has, however, also been criticised for the missing explicit coordination among the intersections and for its simple traffic model that is error-prone in case of long queues [75]. Furthermore, OPAC does not provide facilities for the prioritisation of public transport vehicles.

2.4.4 BALANCE and MOTION

Both adaptive network control systems BALANCE *(Balancing Adaptive Network Control Method*, [74]) and MOTION (*Method for the Optimisation of Traffic Signals in Online-Controlled Networks*, [113, 135]) have their origins at the Technical University of Munich, where they resulted from two EU research projects. Although both systems internally differ and exhibit individual strengths and weaknesses, they follow the same concept: A centralised network-wide optimisation is combined with local traffic-actuated control. On the centralised level, a model of the network-wide traffic demand is created. The model is used to identify the most relevant traffic streams in the network and to optimise the coordination and signal timings of the intersections. The resulting signalisation constraints are then communicated to the network nodes as frame signal plans. The plans ensure the coordination of intersections within the network, but leave some freedom for local adaptations of the signalisation in the case that traffic-actuated node controllers are available.

The working principle of BALANCE and MOTION can be summarised in the following steps:

1. *Traffic detection* Data on the current traffic demand is gathered from the detectors in the network, before it is preprocessed for later use.

2. *Traffic modelling* In a second step, the gathered data is used to create a time-space representation of the current traffic demand in form of meso- or macroscopic traffic models. In the process, missing data and origin-destination relations are estimated, and incidents and congestions are detected. The models are then used to predict the future traffic development and to evaluate control decisions with respect to the resulting delays, stops, and queues.

3. *Optimisation of control parameters* In a third step, the network-wide cycle time, phase durations, phase sequences, as well as offsets are optimised. Both BALANCE and MOTION provide Evolutionary Algorithms to reduce delays and stops that are combined in a single-objective fitness function (see [25] and [135], respectively). In MOTION, evolutionary optimisation is limited to coordination related parameters, i. e., phase sequences and offsets, while all signalisation parameters are considered in BALANCE. Details on the optimisation approaches are given in Section 3.1.3.

4. *Inspection of obtained results* In a last step, a check is performed whether the current signalisation needs to be adapted. If the frame signal plans obtained in Step 3 result in a significant improvement with respect to the objective function, the new signalisation is communicated to the local controllers.

The previous steps are repeated in intervals of a few minutes to achieve an on-line adaptation of the signalisation to the current traffic demand.

Installations of BALANCE and MOTION have been evaluated in several studies: Positive results of a field test of BALANCE at Munich are reported by Friedrich in [74] and – focusing on evolutionary on-line optimisations – by Braun et al. in [25, 26]. For MOTION, Kruse summarises the results of several field tests in [113], while Mück presents results of a recent field test at Münster that applied evolutionary optimisations [135]. Both works report reductions in travel time compared to the systems previously used in the test areas.

The presentation of BALANCE and MOTION completes the discussion of adaptive network control systems in this chapter. In the following, the focus changes from traffic control to traffic modelling. Different types of traffic models are introduced and their specific properties will be outlined.

2.5 Traffic models and simulation

Based on mathematical and logical abstractions of real-world traffic phenomena, traffic models simulate the behaviour of vehicles in a traffic network over space and time. Data on the simulated traffic flows is gathered and can be used to asses the simulated network. Traffic simulations therefore represent experiments performed virtually on a computer rather than in the field. The implicit assumption, however, is that the simulation results carry over to the real-world network despite the abstractions in the model.

Besides their application in adaptive network control systems (Section 2.4), traffic models have several use cases: One example are short term traffic forecasts that support the determination of route guidance recommendations in traveler information systems [207]. Another application is the assessment of transportation planning alternatives, e. g., when there are several options for building a new motorway segment [132]. Furthermore, simulation models can provide estimates on the quality of traffic signal timings (Section 3.1).

For the different application areas, different types of models are in use. In [60], model types are distinguished based on the used representation, the update mechanism, and the handling of randomness in traffic. Regarding representation, simulation models are subdivided in micro-, macro-, and mesoscopic models. *Microscopic models* simulate individual vehicles and their interactions in the traffic network in great detail, while *macroscopic models* take into account the relationships of speed, flow, and density of traffic streams to simulate traffic flow. *Mesoscopic models* combine features of micro- and macroscopic approaches: They simulate individual vehicles, but model their interactions based on macroscopic relationships.

Either type of model keeps track of a system state that represents the network traffic by combining the states of the simulated elements. States

can be updated in fixed or variable intervals, allowing to distinguish between time- and event-based models: *Time-based models* update the system state at fixed time intervals. In *event-based models*, the state update is triggered by simulation events that can occur irregularly. When state changes occur only infrequently, this can lead to run-time reductions. However, traffic models are mostly time-based because of the continuously changing traffic.

Finally, simulation models can be either deterministic or stochastic. *Deterministic models* ignore variabilities in the characteristics of vehicles and their interactions. The models assume, e. g., that all cars have the same length, or that all drivers keep the same minimal distance between their own car and the preceding vehicle when stopped. In *stochastic models*, such properties are determined from statistical distributions using a random number generator. The sequence of generated random numbers depends on an initial numerical value called random seed (or simulation seed). The random seed determines all stochastic decisions during a simulation. Therefore, results from stochastic simulations vary with the random seed for otherwise identical input data. Deterministic models, in contrast, always produce identical results for the same input data.

Overviews on traffic simulation models are provided in various articles (e. g., [24]) and textbooks (e. g., [195]). In the following, traffic models are distinguished by their representation: Section 2.5.1 discusses the basics of microscopic models, Section 2.5.2 is dedicated to macroscopic models.

2.5.1 Microscopic traffic simulation

Microscopic models simulate the characteristics of individual vehicles and their interactions in a traffic network. Simulated vehicles – that exhibit individual properties like a specific length, a predefined acceleration and deceleration capability, or a desired speed – move within a simulated network, where they interact with other vehicles and react on speed limits, traffic signals, and other environmental conditions. The simulation is based on car following and lane changing models that define how vehicles accelerate, decelerate, change lanes, and perform passing manoeuvres. Statistical data on the resulting traffic dynamics (like the average number of stops of a vehicle on a signalised arterial road) can be gathered and

used to asses network properties (like the coordination of traffic signals along the arterial). This section presents key aspects of the simulation process and discusses the calibration of microscopic simulation models.

Simulation model

The core of any microscopic traffic simulator is a simulation model that specifies the analytical relationships and the necessary logic for modelling the traffic flow. A simulation model typically employs several submodels that define, e. g., the car following or the lane changing behaviour of the simulated vehicles. Well known models have been introduced by Gipps [83–85] and Wiedemann [210] and are implemented by the widespread simulators AIMSUN [11, 45] and VISSIM [66], respectively. Instead of presenting one of the several available models in detail, the following discussion will focus on key aspects of microscopic models. It partly follows the presentation of microsimulation modelling software in [60].

Traffic demands At the beginning of a simulation, the simulated traffic network is empty. The network's geometry is modelled by a set of links and intersections and traffic enters the network at predefined origins (or sources). The vehicles traverse the network and leave it again at a set of possible destinations (or sinks). The simulated traffic demand can be either specified by a turning-fraction based or an origin-destination based approach.

Turning-fraction based demand The turning-fraction based approach defines the inflow for each network source and routes vehicles through the network based on turning-fractions that are specified for each intersection. A vehicle is assigned its next turn when it enters a network link based on the turning-fractions specified at the downstream intersection.

Origin-destination based demand In origin-destination based models, the traffic demand is determined by origin-destination matrices (O/D matrices) that specify the flow of vehicles from each origin to each destination. When a vehicle is generated at an origin, it gets assigned a destination where it will exit the network. Since an O/D

pair is usually connected by several routes, a route choice model is used to assign one of the possible routes to a vehicle when it enters the network. Depending on the route choice model, route costs – like distances or travel times – can be calculated once at the beginning of the simulation (fixed route mode) or be updated periodically (variable route mode).

Before the determined routes can be assigned, vehicles need to be created and entered into the network.

Traffic generation When vehicles are created, their attributes need to be defined. Attributes include physical properties of the vehicles (like length or maximal speed) and may also cover aspects of the driver behaviour (like reaction times or gap acceptance parameters). In stochastic simulations, the mean values, deviations, and boundary values of attributes can be defined for each vehicle class (i. e., for cars, buses, or trucks). The particular characteristics for each vehicle are then sampled according to a probability distribution.

After their creation, the vehicles need to enter the network based on the input flows specified by the network's traffic demand. Here, a headway model determines the time interval between two consecutive vehicle arrivals (called headway). Assuming that the specified input flow for an origin is λ vehicles per second according to the current traffic demand, the mean headway equals $1/\lambda$ seconds. Often, an exponential distribution is used to determine the headway h (in seconds) that separates a pair of generated vehicles using the equation

$$h = -\frac{1}{\lambda} \cdot \ln(u),$$

where $u \in]0, 1[$ is a random number. However, the use of other probability distributions or constant headways is possible.

The headway distribution determines the theoretical arrival time for each vehicle. During a simulation run, it is necessary to additionally check whether the arrival is physically feasible or not, i. e., whether there is enough space for the generated vehicle to enter the input link.

Vehicle movement After the vehicles entered the network, they try to travel at their desired speed on the network links. As long as their movement is not influenced by other vehicles, their actual speed is affected by link-specific properties (like the link's slope, its geometry, or its speed limit) only. The simulation model computes the actual speed of a vehicle as the minimum of its desired speed and the speed computed for the link the vehicle is currently traversing.

When a vehicle is influenced by other vehicles in the network, its behaviour is determined by car following and lane changing models. The car following model specifies the interaction between a leader and a follower vehicle traveling in the same lane. Generally, it can be seen as a stimulus-response mechanism where the following vehicle reacts with a specific sensitivity on an observed stimulus, but its reaction is delayed by a reaction time. Most of the employed car following models are fail-safe, i. e., a minimum safety distance between vehicles is maintained. According to [60], the models work in three steps:

1. In a first step, the follower vehicle's speed change in response to the motion of its leader is calculated. The goal is to maintain a target distance between follower and leader that depends on their vehicle characteristics. In general, the acceleration of the follower vehicle is a function of the speeds of the leading and following vehicles, their distance, and the follower's reaction time. Additionally, the acceleration might depend on other parameters specific to a particular car following model.

2. Since the derived acceleration must not exceed the maximum acceleration or desired speed specified for the follower, this is checked in the second step.

3. Finally, the follower's acceleration must guarantee that the follower maintains a minimum separation from the leader. If the car-following acceleration is larger than a safe-following acceleration, the safe-following acceleration is implemented.

Since traffic networks do not consist of single-laned links alone, a lane changing model determines the behaviour of vehicles when they exit their current lane. Whether a lane change can be performed depends on the

gap acceptance of the lane changing vehicle. It will only change lanes if the available gap in the target lane is greater than its critical gap. According to [60], most microscopic simulation models distinguish three types of lane changes:

Mandatory lane changes occur whenever a vehicle must change its lane. Among other reasons, this can be the case when the lane needs to be changed to reach the vehicle's destination or when a lane ends (e. g., an on-ramp on a freeway).

Discretionary lane changes occur when vehicles change their lanes to improve their movement. Lane changes of vehicles that cannot travel at their desired speed because they are hindered by slower moving vehicles in front are a typical example.

Anticipatory lane changes can occur when a vehicle anticipates slowdowns located downstream within its current lane (e. g., due to merges or weaves ahead). In these cases, the decision for a lane change is based on the speed difference of the lanes at the location of the slowdown.

By combining the previously discussed submodels, a microscopic simulation model determines how vehicles move through and interact in the simulated network. Therefore, it has a large influence on the accuracy of the simulation, i. e., it determines how well a simulation run reproduces the traffic conditions observed in the field. A typical microsimulation model comes with a large set of parameters that can – at least partly – be specified by the user of the simulator. The simulation software provides default values for all model parameters, but these will not under all circumstances be able to produce accurate results for a specific network. Therefore, the model parameters need to be calibrated.

Model calibration

The adjustment of model parameters to improve the ability of a simulation model to reproduce the driver behaviour and traffic performance observed in the field is called calibration. Calibration is necessary since even detailed microsimulation models do not include all variables that affect

real-world traffic conditions. Models rely on abstractions of real-world traffic dynamics and can therefore not be expected to work accurately for all possible conditions. Model parameters need to be adapted such that the resulting parameter values minimise an error measure comparing simulated and field traffic conditions.

The details of the calibration procedure are beyond the scope of this thesis, but it should be noted that after calibration microscopic models provide accurate simulations of real-world traffic conditions [19]. Research has been conducted on the automated calibration of microsimulation models using, e. g., the simplex algorithm [105] or Evolutionary Algorithms [206,222] in the process. The continuous calibration of microscopic on-line simulations using Kalman filters has been investigated in [124].

Summary

To summarise the presentation of microscopic simulation models it should be remembered that microscopic models simulate the characteristics of individual vehicles and their interactions in a traffic network in great detail to provide statistical data on the resulting traffic dynamics. Microscopic models are widely applied in a broad variety of use cases since the obtained simulation results are highly accurate after the models have been calibrated. For some applications – like the simulation of traffic-actuated signal control – their use is even indispensable. However, the time required to perform a simulation highly depends on the number of simulated vehicles. For the simulation of large networks, microscopic models that are based on Cellular Automata (like the *Nagel-Schreckenberg Model* [65, 140]) are recommendable due to their comparatively low computational efforts.

2.5.2 Macroscopic traffic simulation

Macroscopic models consider the relationships of aggregated traffic characteristics (i. e., speeds, flows, and densities) when simulating traffic flows. They are based on equations on the conservation of traffic flows and on the propagation of disturbances. Therefore, macroscopic models are well-suited to predict the spatial and temporal development of traffic patterns in a network.

The oldest macroscopic traffic model is the *LWR model* that has been proposed independently by Lighthill and Whitham [121] and Richards [156] and is still popular today. As depicted in Figure 2.7, it describes the collective vehicle dynamics at time t and position x of a road segment in terms of

- the *spatial vehicle density* $\rho(x,t)$,
- the *average velocity* $V(x,t)$, and
- the *traffic flow* $Q(x,t)$.

Regarding the relationship of these terms, the LWR model relies on the basic assumption that flow and velocity are functionally dependent solely on the traffic density.

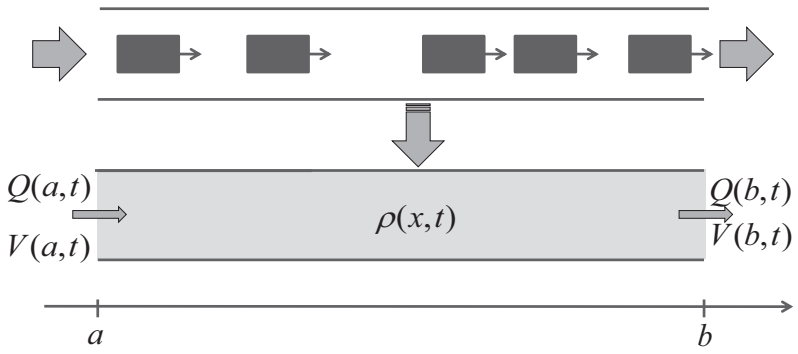


Figure 2.7: Traffic representation in macroscopic models (based on [35])

Theoretical considerations on the relationship of flow, density, and velocity lead to the equation

$$Q(x,t) = V(x,t) \cdot \rho(x,t), \tag{2.6}$$

which states that traffic flow at time t and position x of a road segment can be expressed as the product of density and average velocity. To fulfil the basic assumption of the LWR model, the velocity needs to be expressed as a function f of the density, i. e., $V(x,t) = f(\rho(x,t))$. Reasonably, f should satisfy the following conditions:

- At low traffic densities, drivers are assumed to travel with the maximally allowed speed V_{max}, therefore $f(0) := V_{max}$.
- At the maximal traffic density ρ_{max}, traffic comes to a halt, therefore $f(\rho_{max}) := 0$.
- The velocity is assumed to be monotonically decreasing with increasing densities.

The simplest relation of density and velocity exhibiting the desired properties is a linear function defined by the border conditions $f(0) = V_{max}$ and $f(\rho_{max}) = 0$. It is given by the equation

$$V(x,t) = f(\rho(x,t)) := V_{max} \cdot \left(1 - \frac{\rho(x,t)}{\rho_{max}}\right). \tag{2.7}$$

By inserting this relationship into Equation 2.6, one obtains a quadratic relation for density and flow given by the equation

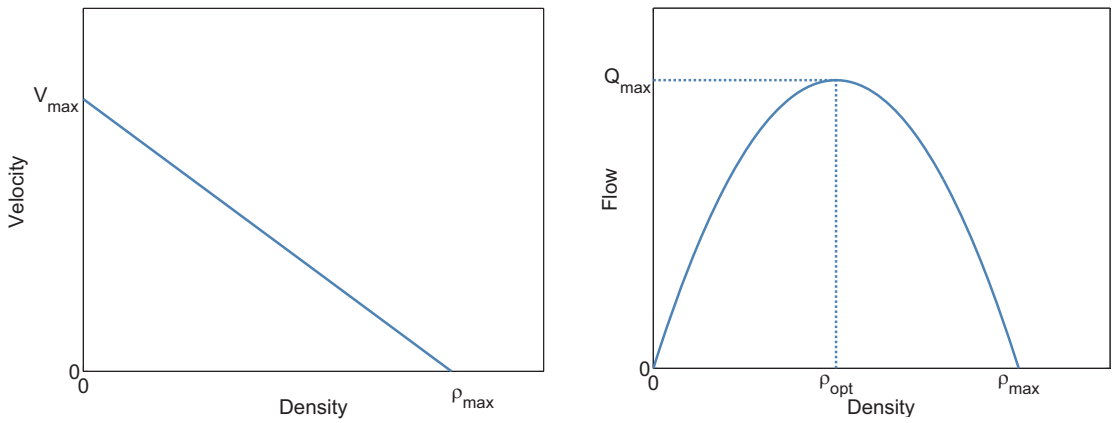
$$Q(x,t) = V_{max} \cdot \left(1 - \frac{\rho(x,t)}{\rho_{max}}\right) \cdot \rho(x,t). \tag{2.8}$$

The relationships of density, velocity, and flow specified in Equations 2.7 and 2.8 are depicted in Figure 2.8. Due to its importance the relation of density and flow is also called *fundamental diagram*. More complex relationships of density and velocity – that better reflect the observations in real traffic networks – can be used in the model. However, they might lead to inconsistencies (like negative speeds or flows for certain densities).

Using Equations 2.7 and 2.8, future traffic developments can be simulated based on the *continuity equation*

$$\frac{\partial \rho(x,t)}{\partial t} + \frac{\partial Q(x,t)}{\partial x} = 0$$

that can be derived since no vehicles get lost within a road segment (see [35] for a detailed derivation). For practical applications, the LWR model is often discretised in time and space (like, e. g., in the cell transmission model [50, 51]).



(a) Relation of density and velocity (b) Relation of density and flow

Figure 2.8: Relation of density, flow, and velocity

2.6 Summary

The chapter initially focused on basic signalisation concepts and defined important terms to be used in the remainder of this thesis. Fixed-time controls that use static signal plans and traffic-actuated controls that adapt their signalisation based on predefined temporal and logical conditions have been introduced. Both approaches to intersection control reflect the expert knowledge of the designing traffic engineer and do not evaluate or optimise their control decisions on-line. In Chapter 5, intersection controls will be extended by an observer/controller architecture that uses methods from machine learning to implement the missing on-line optimisation capabilities.

Following the presentation of basic concepts and control approaches, selected examples of existing adaptive network control systems have been discussed. Adaptive network control systems perform an on-line optimisation of signal plans, thereby typically relying on a centralised architecture and employing traffic models to support the optimisation of control decisions. In contrast to the centralised optimisation of these systems, a self-organising approach to coordination is proposed in this thesis. In Chapters 7 and 8, the approach is introduced and evaluated.

Recapitulating the discussion of traffic models, it should be remembered that different simulation types can be distinguished based on their

representation: Microscopic models simulate individual vehicles and their interactions in the traffic network in great detail, but at the cost of a high computational demand and with the need of parameter calibration. They typically operate stochastically, i. e., the simulation results vary with the random seed for otherwise identical input data. Due to their level of detail, microscopic models are indispensable in the evaluation of traffic-actuated controls. In this thesis, a microscopic simulation model substitutes the real-world road network in the SuOC during the experimental evaluations. Furthermore, the introduced observer/controller framework makes use of microscopic simulations to provide performance estimates in the evolutionary optimisation of signal timings.

Macroscopic models employ a less detailed traffic representation, but allow for the fast simulation of large traffic networks. The models take into account the relationships of speed, flow, and density of traffic streams to simulate traffic flow. Together with mesoscopic models that simulate individual vehicles, but model their interactions based on macroscopic relationships, they are often used in adaptive network control systems or in the network-wide evolutionary optimisation of traffic signals, an application that is discussed in the following chapter.

CHAPTER 3

Optimisation, learning, and self-organisation in traffic control systems

This chapter discusses the optimisation of traffic control systems and presents learning and self-organising control mechanisms:

Optimisation The aim of optimisation is to find optimal solutions for a given problem. In the context of traffic signal systems, optimisation can be used to determine (near-)optimal timing parameters for the signalised intersections in a network. However, optimisations can be time-consuming and their computational requirements limit their applicability in on-line applications with changing traffic demands.

Learning Technical systems that change their behaviour in a way that makes them perform better in the future are said to learn [130, 219]. A learning traffic signal controller might, e. g., adapt the way it switches among the available signal groups based on delay measurements at the intersection. In this example, the delay constitutes a reinforcement signal used for reinforcement learning [186].

Self-organisation A system in which elements interact in order to dynamically achieve a global function or behaviour is called self-organising [63, 81]. The global function or behaviour has to be achieved autonomously as the elements interact with one another. In traffic signal control, self-organising mechanisms are often based on simple predefined rules specifying an intersection's reaction to environmental changes (like an approaching vehicle platoon). The coordination of neighbouring intersections is then obtained by means of sophisticated vehicle detection or communication.

Regarding optimisation approaches, mathematically exact optimisation can be distinguished from heuristics. An approach used for *mathematically exact optimisation* is mixed-integer linear programming [220]. In mixed-integer linear programming, the optimisation problem needs to be formulated as a set of linear equations with constraints which can then be solved by an optimisation software package like CPLEX[1]. The obtained solutions are optimal with respect to the mathematical problem formulation and can – possibly with some quality loss due to abstractions in the model – be carried over to the original problem.

Depending on the mathematical formulation, the time-requirements for optimisations can vary significantly. Köhler et al. used mixed-integer linear programming to optimise offsets in networks of fixed-time controlled intersections [108, 109]. While their initially proposed path-based model required a computing time of nine hours to optimise offsets for a network of nine intersections, an improved link-based approach is reported to require only a few seconds for the optimisation [109].

However, mathematically exact approaches also have drawbacks: The necessary mathematical formulation of the optimisation problem limits their general applicability. Taking the example of [108, 109], the optimisation has been limited to offsets. Other relevant signalisation parameters need to be provided as part of the problem formulation. Furthermore, only fixed-time controls have been considered since traffic-actuated controls are difficult to model.

This thesis focuses on *heuristics* for traffic signal optimisation, since they are more widely applicable. Heuristic approaches include *Evolu-*

[1]IBM Corporation. ILOG CPLEX website: `http://www-01.ibm.com/software/integration/optimization/cplex/`

tionary Algorithms [62], *Artificial Neural Networks* [93], *Particle Swarm Optimisation* [101], and *Ant Colony Optimisation* [59] among others. All mentioned approaches have been applied in transportation engineering, but a comprehensive survey is beyond the scope of this thesis. The interested reader is instead referred to the overviews available in [8, 123, 147, 188, 194].

In the following, Evolutionary Algorithms will be discussed as an example for optimisation heuristics (Section 3.1), while Learning Classifier Systems represent reinforcement learning approaches (Section 3.2). Both mechanisms will later be combined in the organic approach to traffic signal control developed as part of this thesis. Self-organising traffic signals systems are finally reviewed in Section 3.3.

3.1 Evolutionary Algorithms

An *Evolutionary Algorithm* (EA) is a randomised optimisation heuristic that mimics biological evolution to tackle optimisation problems. The working principle of EAs is briefly introduced in Section 3.1.1. Sections 3.1.2 and 3.1.3 present a survey of off-line and on-line applications in traffic signal optimisation, while Section 3.1.4 investigates some general design issues of EAs. Finally, the discussion is summarised in Section 3.1.5.

3.1.1 Working principle

The general scheme of an EA is simple, it is depicted in Figure 3.1: Starting with a set (called *population*) of randomly generated initial solutions, an EA selects solutions with a relatively high quality from its population as parents, which are then combined and locally modified by crossover and mutation operators to form new offspring solutions. Based on their quality, some of the parents and offspring are selected to form the next generation of solutions that replaces the old population. This process is repeated until a stopping criterion (usually a maximum number of generations, a time limit, or some quality level) is reached. Selection, crossover, and mutation are randomised operations, where good solutions have a higher chance to survive and generate offspring. Therefore, the

overall quality of solutions is likely to improve over time while the random influence of mutation helps to prevent premature convergence on some local optimum.

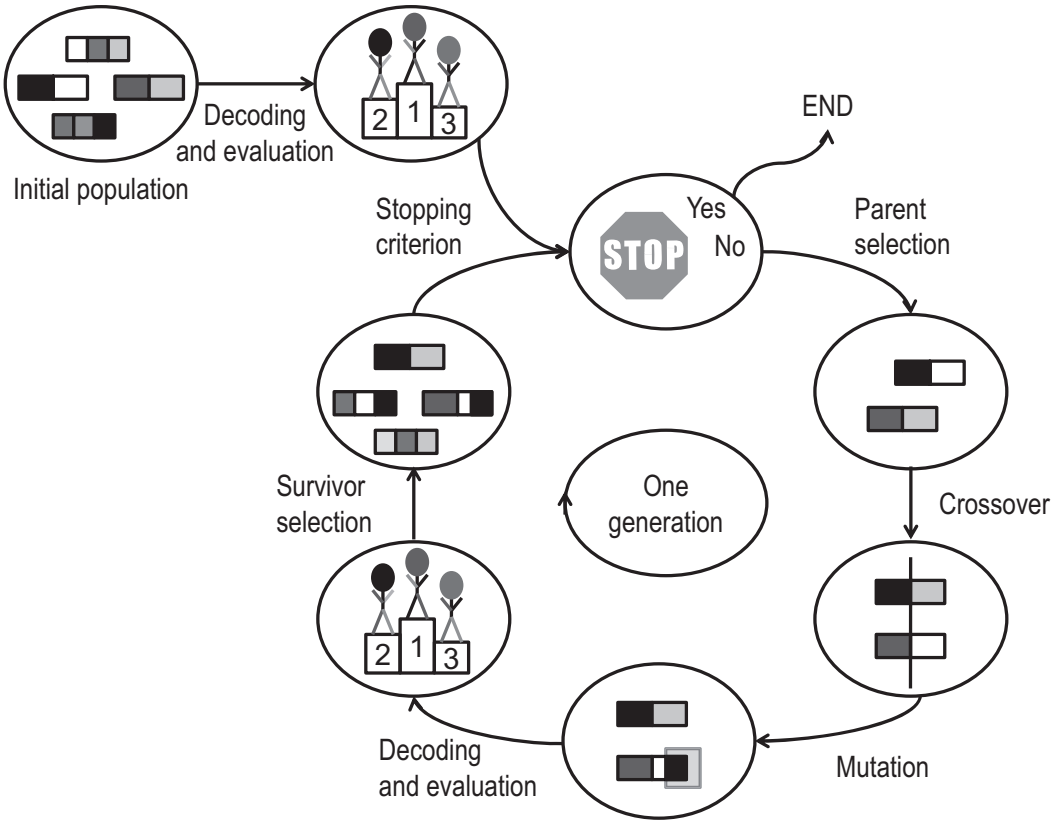


Figure 3.1: Working principle of Evolutionary Algorithms

Although the working principle of EAs is easy to comprehend, their implementation involves several design decisions which are extensively discussed in introductory textbooks like [62, 209]. Here, selected aspects are briefly addressed:

Representation A first necessary step is to design a computer representation (called *genotype*) that represents possible solutions for the real-world problem (the *phenotypes*) in the evolutionary process. While selection, crossover, and mutation work on the genotype, the phenotype is needed for fitness evaluations (see Figure 3.1; genotypes are depicted by boxes while phenotypes are shown as stickmen).

Evaluation function The evaluation (or fitness/objective) function forms the basis for selection and thereby guides the evolutionary search. Since evaluations are performed frequently, functions that need a large amount of computation time can significantly slow down the search. Stochastic fitness functions that assign noisy fitness values to a solution (like microscopic traffic simulations) can be used, but require special care [169].

Selection mechanism Two selection mechanisms are working within the general EA scheme. *Parent selection* is used to determine those solutions from the population that undergo variation (crossover or mutation) in order to create offspring. *Survivor selection* determines which solutions will be included in the next generation's population. Both mechanisms distinguish among solutions based on their quality to allow the better individuals to reproduce and survive. The choice of selection mechanisms influences the *selection pressure*. A mechanism that strongly favours high quality solutions results in a high selection pressure which might lead to premature convergence of the search. On the other hand, searches involving a selection pressure that is too small will typically require a long time to find near-optimal solutions.

Variation operators New candidate solutions are created from solutions contained in the current population using variation operators. *Crossover* (or *recombination*) is an operator that merges information from (typically) two parents in one or two offspring solutions. Crossover works stochastically, i. e., it randomly determines which parts of the parents are merged. The idea is to combine the desirable features of the parents in the offspring solutions. In contrast to crossover, *mutation* is applied to a single parent and delivers a (slightly) modified mutant depending on the outcome of a series of random choices.

Further aspects The design of EAs involves further aspects including the *population size* or a *stopping criterion*. Like the choice of representation, selection mechanisms, and variation operators, their specification largely depends on the optimisation problem.

Historically, different flavours of EAs were developed by different researchers. In the 1960s, Rechenberg and Schwefel invented *Evolution Strategies* [154, 173], while Holland presented *Genetic Algorithms* [96]. Both approaches tackle optimisation problems, but differ in the genetic representations and operators used: While Evolution Strategies typically use a real-valued parameter representation and primarily rely on mutation to create offspring, Genetic Algorithms work with a binary coding and give a stronger emphasis on recombination in their search.

Other EA variants aim at the automatic generation of computer programs: *Evolutionary Programming* (developed in the 1960s by Fogel et al. [68]) and *Genetic Programming* (invented by Koza in the 1990s [110, 111]) investigate the evolution of finite state machines or tree structures, respectively. In agreement with contemporary terminology, Evolution Strategies, Genetic Algorithms, Evolutionary Programming, and Genetic Programming are subsumed under the term *Evolutionary Algorithm* in this thesis.

Still today, EAs are an active area of research with several dedicated conferences, including the *Congress on Evolutionary Computation* (CEC), the *Genetic and Evolutionary Computation Conference* (GECCO), and the *Conference on Parallel Problem Solving From Nature* (PPSN). Furthermore, journals like *Evolutionary Computation* or the *IEEE Transactions on Evolutionary Computation* are addressing the field.

In the following, several applications of EAs to off-line and on-line traffic signal optimisation are presented.

3.1.2 Off-line optimisation of traffic control systems

The first work known to the author that used EAs for signal timing determination has been published by Foy et al. in 1992 [71]. In a simulated Manhattan-type network of four two-phased intersections, cycle length and green time splits are optimised for a fixed traffic demand. The minimisation of the resulting vehicular delay serves as objective. According to Foy et al., their EA finds near-optimal solutions which proves the feasibility of EAs for the task. In the following, a selection of more recent approaches to evolutionary traffic signal optimisation is presented.

Isolated intersections

In her dissertation [203], Vogel investigates the off-line optimisation of a fixed-time controlled intersection. In contrast to other authors, Vogel does not restrict her work to the optimisation of cycle time and phase durations, but includes the phase sequence and system in the evolutionary process: The EA combines the available signal groups into a sequence of phases before cycle time and phase durations are specified.

Since a candidate phase system cannot be evaluated independently of the remaining parameters of a controller, Vogel uses two nested evolutionary loops: In an outer loop, new candidate phase systems are created by applying genetic operators (crossover and/or mutation) on the solutions stemming from the loop's previous iteration. Each of the candidate phase systems is then treated in an inner optimisation loop that aims at finding the best cycle time and phase durations for the candidate. Once the inner loop is finished, the resulting signal plan is evaluated using an event-based simulation software. Based on the evaluation results, the surviving phase systems – which serve as parent solutions in the next iteration of the outer loop – are selected and an iteration of the outer loop is completed.

Vogel does not explicitly compare the evolved phase systems to a reference solution, but she reports that the evolutionary approach yields "reasonable" results. However, experiments that optimised only cycle and phase durations of the intersection were compared to two-, three-, and four-phased signal plans created according to Webster's method (see Section 2.2.2) and are reported to result in the same or a reduced average delay.

Network-wide optimisation

While Vogel restricts her work to a single intersection, Braun et al. developed an evolutionary approach called GALOP (*Genetischer Algorithmus zur Lichtsignaloptimierung in städtischen Netzen*) that focuses on the network-wide optimisation of coordinated traffic lights [27–29]. GALOP has been incorporated in the traffic engineering workstation CROSSIG to allow for a seamless integration with established planning tools. While a traffic engineer can specify constraints for the optimisation (like allowed

phase sequences or minimum green times), GALOP evolves feasible signal plans specifying the network-wide cycle time as well as the phase sequences, phase durations, and offsets for the network's intersections. For constraint handling, GALOP relies on a relative coding of parameters. Whenever possible, the genetic representation of a signal plan does not contain absolute parameter values, but represents a parameter's value relative to the given constraints: Instead of specifying, e. g., the cycle time t_C by its duration in seconds, it is specified as a fraction φ, $0 \leq \varphi \leq 1$, of the difference between the maximal feasible cycle time $t_{C,max}$ and the minimal feasible cycle time $t_{C,min}$. The cycle time is then obtained by calculating $t_C = t_{C,min} + \varphi \cdot (t_{C,max} - t_{C,min})$. Details on the representation used within GALOP are presented in [28, 29].

Since network-wide optimisations consider a large number of parameters, a fast simulation model is essential to allow for the evaluation of a reasonable number of solutions within an acceptable amount of time. GALOP uses the macroscopic flow model of the traffic control system BALANCE (see Section 2.4.4) which has been designed to satisfy the strict time restrictions associated with adaptive network control systems. As optimisation criterion, GALOP handles a single-objective fitness function.

Signal plans evolved with GALOP have been evaluated in a field test [27]. The test site included six intersections located at Regensburg, Germany, which were controlled by three time-dependent signal plans handling the morning, day, and evening traffic, respectively. The signal plans used in the field were compared to solutions evolved by GALOP with respect to the resulting travel time and the induced number of stops. Comparisons were based on floating car data and travel time measurements stemming from vehicle re-identification. Braun et al. report that solutions evolved by GALOP improve the morning and evening programme especially with respect to the number of stops, while GALOP's day programme showed a similar performance to the signal plan used in the field.

Parallel evaluations

The run-time of EAs can grow large when fitness evaluations are time-consuming (e. g., because they are based upon traffic simulations). For-

tunately, candidate solutions in an EA's population can be evaluated independently, thereby allowing for a parallelisation of the search process using a *master-slave EA* [5, 42]: While one computer – the master – keeps track of the population and performs selection, mutation, and crossover, several slave computers (or processors/cores of a parallel computer) run the time-consuming fitness evaluations in parallel.

At the cost of additional computing power, a master-slave EA can significantly reduce the run-time needed for an optimisation. However, an efficient parallelisation in terms of invested resources depends on a balanced distribution of workload among the participating computers and on a small overhead for data exchange. In [89], Girianna and Benekohal investigate the application of a master-slave EA for traffic scenarios and analyse the run-time and efficiency of the parallel approach for different numbers of processors and several network sizes.

Several authors rely on parallel fitness evaluations to speed up their heuristic search: Sánchez et al. started their work on evolutionary approaches for the off-line optimisation of traffic networks in 2004 by investigating an artificial network of four intersections [162]. An EA is used to evolve fixed-time signal plans for the network's intersections based on a predefined cycle time. The cycle is split into equally-sized fractions and the EA's task is the network-wide selection of active signals for all resulting time slots. To evaluate the quality of evolved solutions (that is measured by the vehicles' mean time at the network), a deterministic Cellular Automaton-based simulation model is used. The parallel evaluation of candidate solutions relies on a master-slave EA that runs on a computer cluster of five machines.

Recent works of Sánchez et al. still rely on the deterministic simulator and on the parallel evaluation of solutions, but investigate simulations of larger real-world networks [165, 166]. The developed optimisation process takes into account the phase durations (and as a result the cycle times) of fixed-time controlled intersections, but it does no longer consider their phase system or sequence. In contrast to other authors dealing with network-wide optimisations, Sánchez et al. do not explicitly optimise the coordinated operation of neighbouring intersections. No common cycle time for (parts of) the network is enforced in the optimisation process and no explicit offset optimisation takes place. However, a coordinated

operation of neighbouring nodes can result from the optimisation of the intersections' phase durations.

As test case, a simulation model of a traffic network consisting of eleven intersections located at Santa Cruz de Teneriffe, Spain, is investigated. The number of vehicles that left the network during the simulation serves as single-objective fitness function. Results show that with the evolved traffic light timings up to 26 % more vehicles could leave the network compared to the solutions supplied by the City Council.

In [181], Stevanovic et al. present VISGAOST, a *VISSIM-based Genetic Algorithm Optimisation of Signal Timings*. VISGAOST is capable of optimising cycle time, phase sequence, phase durations, and offsets for a network of coordinated traffic-actuated NEMA controllers [141]. The microscopic traffic simulation software VISSIM [66] is used to evaluate the quality of the evolved candidate signal plans. Since microscopic simulations are time-consuming (especially when large networks have to be simulated numerous times), VISGAOST relies on a master-slave EA for parallel fitness evaluations.

A specialty of VISGAOST is that not all parameters of a signal plan are optimised at the same time. The optimisation process is split into intervals which are dedicated to the optimisation of parameter subsets: While some of the intervals might only consider cycle time and offsets and keep all remaining parameters fixed, other intervals allow the variation of other parameters or of the complete signal plan. This partial optimisation is reported to yield better results compared to runs which always varied all defining parameters of a signal plan.

VISGAOST has been evaluated in a simulation study for an arterial road network of twelve intersections in Park City, USA. The network was controlled by coordinated traffic-actuated NEMA controllers. The signal plan used in the field and a solution found by SYNCHRO[2] – a traditional optimisation tool – served as reference solutions. The signal plan found by VISGAOST could outperform the reference solution by at least 8 % with respect to a single-objective fitness function that combines network-wide delays and stops. However, the run-time required by VISGAOST using nine computers running slave processes was 90 hours.

[2]Trafficware Ltd. SYNCHRO website: `http://www.trafficware.com/`

Recently, VISGAOST has been extended to additionally optimise transit priority settings that allow for a preferential treatment of public transport vehicles [182]. The additional priority settings specify the extension of green phases with approaching public transport vehicles as well as the truncation of preceding phases to allow for a faster service of priority vehicles approaching a red light. The transit priority settings can be optimised separately or in combination with basic signal settings. The approach has been evaluated for an arterial of seven intersections at Albany, USA. Results show that the initial solution found by SYNCHRO could be improved by VISGAOST with respect to the average delay per traveller.

Multi-objective optimisation

The previous references use a single-objective fitness function, even if several objectives (like delays and stops) are considered in the optimisation. Instead of treating each objective separately, all objectives are combined in a single evaluation function using weights to determine their relative importance. Assuming conflicting objectives, the drawback of this approach is that only a single optimal trade-off among the objectives can be found, although several equally optimal other trade-off solutions exist: In the case of more than one conflicting objective, a problem has no single optimal solution, but a set of solutions which are all *Pareto-optimal*. These solutions cannot be improved in any objective without worsening at least one other objective (see Figure 3.2 for an illustrating example).

For multi-objective problems, *multi-objective EAs* have been developed [55]. Instead of combining the different objectives into a single fitness function and searching for a single optimal solution afterwards, multi-objective EAs search for a set of Pareto-optimal solutions. In their search, they rely on the concept of domination: A solution x is said to *dominate* a solution y, if x is no worse than y in all objectives and if x is strictly better than y in at least one objective. Solutions that are non-dominated are favoured by the selection mechanism of the algorithm and thereby guide the evolutionary search. Well-known multi-objective EAs are NSGA-II [56], ParEGO [107], and SPEA2 [223].

Applications of multi-objective evolutionary optimisation to traffic signal systems are discussed by Sun et al., Branke et al., and Schmöcker

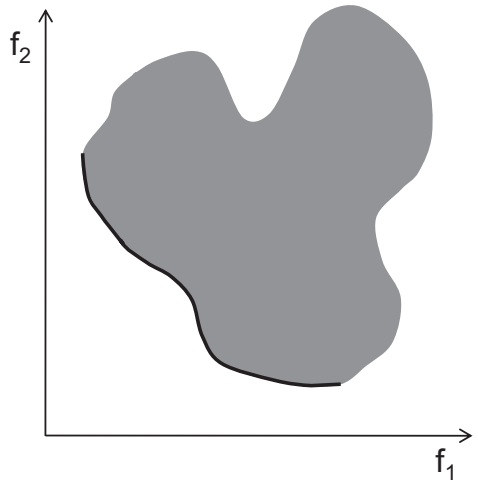


Figure 3.2: Pareto-optimal solutions (marked black) for a two-objective minimisation problem (based on [55])

et al. among others: Sun et al. investigated the use of NSGA-II for signal timing optimisation [185]. Delay times and stops were minimised for a two-phased fixed-time controlled intersection. NSGA-II optimised the phase durations based on fitness values calculated by approximation formulas for delay (Webster [208]) and stops (Akçelik [4]). Sun et al. conclude that multi-objective EAs have the potential to be used for traffic signal timing optimisation.

Branke et al. optimised the operation of a VS-Plus[3] controlled intersection at Karlsruhe, Germany [22]. Several parameters related to the intersection's traffic-actuated operation are considered in a multi-objective optimisation using NSGA-II. Objectives include combinations of vehicular delay, pedestrian delay, and vehicular stops. In contrast to Sun et al., the evaluation of evolved solutions relies on results of the microscopic simulation software VISSIM [66] instead of approximation formulas. Solutions found by NSGA-II outperformed a reference solution provided by a traffic-engineer with respect to the considered objectives. However, results indicate the importance of explicitly including all relevant criteria as objectives, since otherwise evolved solutions tend to be good with respect to the objectives, but at the cost of other unconsidered criteria.

[3]Verkehrs-Systeme AG. VS-Plus website: `http://www.vs-plus.com/`

Schmöcker et al. investigated the multi-objective optimisation of membership functions for a fuzzy logic signal controller [170]. The fuzzy logic controller adapts an intersection's phase durations based on local vehicle queues and pedestrian delays. Its fuzzy rule base is predefined, but the membership functions used for fuzzification and defuzzification are subject to optimisation. Instead of using a multi-objective EA, Schmöcker et al. expect the designing traffic engineer to specify acceptability and unacceptability thresholds for each objective. Based on these thresholds, an EA searches for the Bellman-Zadeh optimal solution [18] that maximises the minimum acceptability with respect to all objectives. The Bellman-Zadeh optimal solution is proven to be Pareto-optimal, while being located in the centre of the Pareto-optimal set.

The approach has been evaluated for an intersection that is part of a SCOOT controlled arterial (see Section 2.4.1) located at London, UK. The intersection's phase durations have been adapted considering local vehicular queues and pedestrian delays as well as network-wide delays for different vehicle classes. Compared to the intersection under SCOOT control, the priorities for public transport or pedestrians could be improved at the cost of an increased vehicular delay.

3.1.3 On-line optimisation of traffic control systems

In all previous references, EAs have been developed to be applied off-line at the design stage of a traffic signal system. At design time, signal settings are optimised considering a few typical traffic demands occurring during the day (like morning or evening peaks), only. Therefore, run-times of several hours for the necessary optimisations can be acceptable. However, modern adaptive network control systems (see Section 2.4) continuously adapt and optimise a network's signalisation on-line, thereby allowing for a flexible reaction to changing and unforeseen traffic demands. While continuous optimisations can improve the performance of traffic signal systems, the acceptable run-time for a single optimisation is drastically reduced to a few minutes. Therefore, the on-line use of EAs has only been investigated recently.

In evolutionary on-line optimisations, it is especially important to speed up the fitness evaluations: In complex fitness landscapes, EAs

typically need a large number of evaluations to evolve reasonable solutions. Whenever these evaluations are time-consuming – which is the case for traffic simulations – this results in a relatively large running time. Furthermore, EAs cannot formally guarantee a minimum quality for evolved solutions, although the expected quality can be good. Several approaches that tackle these challenges are presented in the remainder of this section.

Macroscopic evaluations

Almasri and Friedrich investigated the evolutionary optimisation of offsets in signalised traffic networks [6, 7, 76, 77]. A special focus is given to the time requirements of their approach to make it suitable for online applications. Almasri and Friedrich relied on the cell transmission model [50,51] for evaluating the solutions evolved by their EA. The model follows a macroscopic approach, but is discretised in time and space. It has been reported to combine accurate results with limited computing time requirements.

Two slightly different EA variants were used to optimise the offsets of signalised fixed-time controlled intersections with the objective of reduced network-wide delay. Both EAs implemented a binary coding of offset values, but their variation operators either modified the offset of single intersections (in a predefined sequence that incorporated traffic engineering experience) or varied offset values of several intersections simultaneously.

The two investigated test cases were an arterial of three intersections and a subnetwork of six intersections that is located at Hannover, Germany. For the arterial network, both EAs were capable of evolving the optimal solution (that has been determined using full enumeration) in less than 100 CPU seconds. In comparison to offsets obtained using the dominance method from engineering practice [171], a reduction of 10 % could be obtained. For the larger network, near optimal solutions were obtained, but the run-time was significantly increased. Therefore, Almasri and Friedrich report that "the limits of online adaptation are already reached for the comparatively small test network" [7].

Girianna and Benekohal applied a master-slave EA for signal coordination in networks with oversaturated intersections [86–88, 90]. In over-

saturated traffic conditions, vehicle queues fill intersection approaches and thereby interfere with the operation of neighbouring upstream traffic lights. The removal of queues and blockages becomes especially important in these conditions. In [90], Girianna and Benekohal used an EA to calculate the future phase durations for two-phased fixed-time controlled intersections in an oversaturated grid network. The network was assumed to have constant entry flows and consisted of 20 intersections. A solution specified the phase durations (and as a result cycle times and offsets) of all intersections in the network for several cycles in advance. The future traffic development was determined by a set of equations that allow for the calculation of flows and queues resulting from a candidate solution. The objective was to maximise the number of released vehicles while violated constraints (like queues exceeding the storage capacity of a link) were included as penalties. The EA is reported to generate signal timings that provide a sound handling of queues within the network.

Integration within adaptive network control systems

Recently, EAs have been integrated in the adaptive network control systems BALANCE and MOTION (see Section 2.4.4):

BALANCE Based on the GALOP approach for the network-wide off-line optimisation of traffic signals (see Section 3.1.2), Braun et al. developed an evolutionary on-line system within the research project TRAVOLUTION [25, 26]. An EA optimises a frame signal plan that specifies the network-wide cycle time as well as intersection-specific offsets, phase sequences, and time frames bounding possible phase endings. Based on the frame signal plan, local traffic-actuated controllers can adapt the green times at each intersection within the specified time frames.

In TRAVOLUTION, optimisation minimises a single-objective problem that aggregates the vehicular delay at all intersections. Evolved frame signal plans are evaluated based on the flow model used within BALANCE. The model consists of meso- and macroscopic components and has been specifically designed for fast and accurate simulations, thereby making a network-wide on-line optimisation feasible.

For evaluation, a field test at Ingolstadt, Germany, has been conducted which included 46 intersections within the city's main road network.

For optimisation purposes, the intersections were grouped into three subnetworks, so the EA can tackle the tasks within the existing time-restrictions. The traffic model of each subnetwork is updated on-line based on local traffic measurements reported from the intersections. Additionally, the current traffic flows in the neighbouring subnetworks are considered.

Evolved frame signal plans were compared to a basic scenario having only local actuated control and to frame signal plans optimised by a hill climber that is part of BALANCE. Floating car data (that includes GPS-based location information and additional data from the vehicles' CAN-bus) and automatic vehicle re-identification were used during the field test to evaluate the vehicular delay and the number of stops. Using the EA, the delay could be reduced by 21 % compared to the basic scenario and by 10 % compared to the hill climber. The number of stops was reduced by 17 % and 8 %, respectively.

MOTION While Braun et al. integrated their EA in the adaptive network control system BALANCE, Mück discusses the application of an EA within the MOTION control system [135]. A self-adaptive Evolution Strategy has been added to MOTION to optimise the traffic light coordination in the network. The Evolution Strategy simultaneously optimises offsets and phase sequences for the network's intersections, selecting the phase sequence for each intersection from predefined sets of applicable sequences. Optimisation objective is the reduction of delays and stops that are evaluated using a single-objective fitness function. Fitness values are obtained from a mesoscopic traffic model. Due to the time-restriction in on-line applications, Mück reports that solutions calculated on previous days are included in the initial population whenever they are available. Constraint violations are handled by a repair operator.

Results are reported for a field test at Münster, Germany, that involved an arterial road with 13 intersections. Using the evolutionary approach, an on-line optimisation of the coordination could be performed within the available time frame of two minutes. Mück estimates that the optimisation of networks of up to 30 or 40 intersections should be manageable within this time limit. Solutions found were as good or better compared to a finely tuned reference solution that was previously used in the field,

although the parameterisation of the algorithm was basically limited to weights of the fitness function.

Despite the positive results, Mück expresses some critical comments regarding the described evolutionary approach:

Goals of a planner vs. the system optimum For the test field at Münster it was observed that vehicles turning onto the arterial benefited from the evolved coordination while a fraction of the vehicles that travelled the arterial from the beginning had to suffer disadvantages. Although this coordination might be close to the system optimum, this is usually not intended by human planners since it leads to discomfort for commuters. According to Mück, an improved traffic model – that allows to distinguish the two types of travellers – could help to solve this drawback.

Robustness of solutions When optimising the coordination within a road network, there are typically several nearly optimal solutions of a similar fitness. These solutions often exhibit different coordination structures. Looking at the example of an arterial road with approximately the same amount of traffic travelling in both directions, a coordination in either direction would result in solutions of a similar quality. In such cases, it can depend on small changes in traffic which solution has the better fitness. As a result, a frequent change of coordination might occur that has a negative impact on traffic. The problem is also reported by Braun [25]. A solution to this problem might be an additional check that verifies whether the new coordination results in a significant improvement and discards it otherwise.

Reproducibility of results When using adaptive network control systems in the field, it is important that their behaviour can be understood, tracked, and reproduced – especially in case of complaints of road users. Evolutionary optimisation is a random process that returns different solutions when repeatedly performed. Nevertheless, optimisations can be reproduced when important inputs (like the random seed) are saved. However, guarantees for a certain quality of obtained solutions are generally hard to give.

Despite the criticism, Mück reports that the use of EAs within MOTION will be investigated further.

3.1.4 Design issues in evolutionary traffic signal optimisation

In the previous subsections, several applications of EAs for traffic signal optimisation have been presented. The different approaches investigate the single- or multi-objective optimisation of fixed-time controlled or traffic-actuated intersections as well as the coordination of signals within traffic networks. Although numerous approaches reached or improved the quality of sophisticated reference solutions, it remains difficult to deduce general guidelines for successful EA design since the presented approaches rely on different representations and use a wide variety of selection and variation operators. Nevertheless, some works explicitly focus on design aspects in evolutionary traffic signal optimisation. These will be briefly discussed in the following.

Number of generations and population size

The appropriate choice of the number of generations and the population size in traffic signal optimisations has been experimentally investigated by Abu-Lebdeh and Al-Omari [2] and by Abu-Lebdeh and Benekohal [3], respectively. Both works use a *micro-genetic algorithm* (micro-GA) that relies on a small-sized population of binary coded individuals which quickly converges to a local optimum. Whenever this happens, the micro-GA replaces large parts of the population by random individuals to ensure diversity.

In [3], the influence of the *population size* on the performance of a micro-GA has been experimentally investigated for a traffic scenario. Among other aspects, the fitness that was reached after a different number of evaluations and the dispersion of the evolved solutions' fitness around the mean served as indicators for a good population size. Based on the obtained results, Abu-Lebdeh and Benekohal propose heuristic approaches to experimentally determine the optimal population size in preliminary tests. As a rule of thumb, a population size of $\sqrt{L}$ – where L

is the string length of the binary coded individuals – is proposed whenever sufficient computational resources are available.

With respect to computational resources, Abu-Lebdeh and Al-Omari investigate the *number of generations* necessary to optimise several variants of a traffic control problem that differ in size and complexity [2]. They propose to use the string length L of the binary-coded individuals as lower bound for the number of generations of a micro-GA while the population size is determined as $\sqrt{L}$. Although the authors conclude that this choice worked well for their traffic scenario, it could not be verified for a deceptive problem with many local optima. To what extent the recommendations carry over to different algorithm and problem types remains unclear.

Representation

Sánchez et al. compared crossover operators for binary and real-valued representations of signal timings within a traffic network [163]. The two representations coded the phase durations of the network's traffic lights as a sequence of real numbers or their binary Gray codes, respectively. To create new individuals, the crossover operators split the representation of two parents into parts and exchange the resulting sub-sequences. Since for the binary representation splits can occur within the Gray code of a phase duration, the operator combines and modifies the phase durations of the parents while for the real-valued coding only a recombination takes place. Sánchez et al. report that in their experiments the binary crossover (in combination with a variable mutation probability) outperformed the real-valued alternative.

The findings of Sánchez et al. contradict the results of Michalewicz, who reports for a multi-dimensional numerical problem that "the floating point representation is faster, more consistent from run to run, and provides a higher precision" [128]. Therefore, other researchers (like Braun [25]) rely on real-valued representations when optimising traffic signal timings.

Fitness evaluation

The handling of stochastic microscopic simulations for fitness evaluation has been investigated by Kesur [104] and Sánchez et al. [164]. Due to

the stochastic nature of vehicle arrivals and random driver behaviour, a microscopic simulator provides slightly different results each time a particular signal plan is evaluated (see Section 2.5.1). For an EA, this results in *noisy fitness evaluations* that are perturbed by a random error term with zero mean. Kesur [104] investigated several techniques to reduce the noise in stochastic fitness evaluations:

Repeated simulations By performing repeated simulations of the same signal plan and using the mean result as fitness, the signal plan's "true" fitness can be estimated well. However, there is a trade-off when simulations are time-consuming and the computing budget is limited since an increased number of simulations per fitness evaluation decreases the extent of the evolutionary search.

Generational reevaluation When evaluating signal plans based on a single simulation only, a signal plan might receive a "lucky" evaluation and will therefore be favoured by selection although its true fitness is poor. To resolve this problem, individuals which survive unaltered from one generation to the next can be reevaluated at the cost of additional simulations. Kesur investigated whether reevaluations have a positive effect on the search efficiency.

Common random numbers A comparison of different signal plans can be based on simulation runs that use common random numbers to ensure the identical outcome of random choices within the simulations. As Rathi [152] reports, microscopic simulations that are based on common random numbers decrease the likelihood of drawing false conclusions when comparing signal plans. Therefore, the potential benefits of common random numbers have also been examined by Kesur.

In his experiments, Kesur investigated three EA variants for the optimisation of traffic signals in two test networks. Both undersatured and oversaturated traffic conditions were considered since microscopic simulations exhibit a larger variability in oversatured conditions. Results indicate that the generational reevaluation does not provide any measurable benefit. Furthermore, a single simulation with common random

numbers is found to be optimal for fitness evaluations even in oversaturated conditions. These findings are in compliance with the results of Sánchez et al. [164] who also propose to use common random numbers and avoid time-consuming repeated stochastic simulations for fitness evaluations.

Design methodology

Regarding the design and configuration of EAs, some recommendations available in the literature have been discussed above. However, their general applicability remains questionable since some of the findings are contradictory and it remains unclear whether reported results will carry over to other algorithm or problem types. Therefore, an application-specific configuration of EAs remains indispensable. To this end, general design methodologies that involve iterated preliminary tests are proposed in [209].

3.1.5 Summary

EAs are nature-inspired optimisation heuristics that mimic biological evolution to tackle optimisation problems. Since their first application to traffic signal optimisation in the beginning of the 1990s, several researchers from different fields of science have investigated the evolutionary optimisation of traffic signals. Initially, the focus was on off-line applications, i. e., EAs were used at the design time of a signal system. Published works focused on single intersections as well as networks, used multi-objective optimisation to simultaneously optimise conflicting objectives, and investigated parallel approaches to speed up the often time-consuming optimisation process. Only recently, the focus of the research community shifted to on-line scenarios, where EAs optimise the signalisation at run-time. In these scenarios, the time required for an optimisation is a critical factor, since fast optimisations are a prerequisite for a quick adaptation to changing traffic demands. The necessary speed-ups have been mainly obtained by the use of fast meso- or macroscopic simulation models. Sections 3.1.2 and 3.1.3 discussed selected examples of evolutionary optimisation in off-line and on-line scenarios which are summarised in the upper and lower half of Table 3.1, respectively. However, the

review presented here is not comprehensive. More extensive overviews are available in [1, 123, 147].

Most of the published results report EAs to be competitive to traditional optimisation approaches or hand-made signal plans, often indicating that evolved results could even outperform the former reference solutions. An advantage of EAs over other optimisation approaches lies in their general applicability to problems where a fitness function for evaluating a solution's quality is available. In traffic engineering – where sophisticated simulation models are at hand (see Section 2.5) – EAs can be easily applied to a wide variety of problems. In contrast, exact optimisation approaches require a complex mathematical problem formulation to be applicable.

Another advantage of EAs is that optimisations can be easily accelerated by parallelisation: The fitness evaluations – which are typically responsible for the largest part of an EA's run-time – can be distributed among several machines. Furthermore, multiple conflicting objectives can be simultaneously handled by multi-objective EAs that evolve a set of non-dominated trade-off solutions.

Despite the mentioned advantages, the application of EAs is not without pitfalls. Although their working principle is easy to understand, a problem-specific algorithm design can be challenging. The literature provides guidelines for the design of EAs [209], but general recommendations for specific problem classes are rare and preliminary tests are unavoidable.

Further drawbacks that hinder the application of EAs in on-line scenarios are their run-time requirements in complex search spaces and missing quality guarantees for the outcome of optimisations: Solutions to the run-time challenge include fast simulation models (like those used in Section 3.1.3) or parallelisation (see Section 3.1.2). The missing quality guarantees can be tackled by a combination of EAs and other machine learning techniques (see Chapter 5). The already large amount of successful traffic signal optimisations indicates that these challenges can be successfully handled in the future. Therefore, EAs are a valuable tool that can support traffic engineers in the design and optimisation of traffic signal systems. Leaving the area of evolutionary optimisation, the following section focuses Learning Classifier Systems which are a widely used reinforcement learning mechanism.

Table 3.1: Evolutionary traffic signal optimisation

Author	Test case[a]		Controller/parameters					Objectives				
	Intersection	Network	Controller type[b]	Phase sequence	Phase duration	Cycle time	Offset	No. of objectives	Evaluation[c]	Time[d]	Stops	Other
Branke et al. [22]	s		a	(x)	x	(x)		2	m	x	x	
Braun et al. [27–29]		f	f	x	x	x	x	1	M	x	x	
Foy et al. [71]		s	f		x	x	(x)	1	m	x		
Sánchez et al. [162]		s	f	x	x		(x)	1	m	x		
Sánchez et al. [165, 166]		s	f		x	(x)	(x)	1	m			x
Schmöcker et al. [170]	s		fuzzy		x			5	m	x		x
Stevanovic et al. [181, 182]		s	a	x	x	x	x	1	m	x	x	
Sun et al. [185]	s		f		x	x		2	M	x	x	
Vogel [203]	s		f	x	x	x		1	m	x		
Almasri et al. [6, 7, 76, 77]		s	f				x	1	M	x		
Braun et al. [26]		f	a	x	x	x	x	1	M	x		
Girianna et al. [90]		s	f		x	(x)	(x)	1	M			x
Mück [135]		f	a	x			x	1	M	x	x	

[a] Field test (f) or simulation study (s)
[b] Fixed-time (f) or traffic-actuated (a)
[c] Microscopic (m) or meso-/macroscopic (M) simulation, approximation formula (a)
[d] Travel, delay, or waiting time

3.2 Learning Classifier Systems

A *Learning Classifier System* (LCS) is a rule-based learning system that combines techniques from reinforcement learning [186] with ideas of EAs. LCSs learn problem-specific knowledge that is stored in a set of classifiers that is called *population* or *rule base*. A *classifier* is a condition-action-prediction rule representing a small portion of the acquired knowledge. Its condition specifies a part of the problem domain to which the classifier is applicable, its action identifies a possible solution, and the prediction gives an estimate on the action's quality considering the identified subproblem.

Classifiers are constantly updated in the learning process: Evolutionary operators are used for modifications that aim at an adequate decomposition of the problem domain into a set of subproblems for which appropriate actions are known. Reinforcement mechanisms update the classifiers' prediction estimates based on received rewards.

Considering their working principle, Michigan and Pittsburgh Classifier Systems can be distinguished:

Michigan Classifier Systems Using the term "cognitive system", Holland [95] laid the foundations for Michigan Classifier Systems and presented the first detailed LCS description in collaboration with Reitman [97]. Michigan Classifier Systems are designed for *on-line learning*. They keep track of a single classifier population that represents the current knowledge of the LCS. An evaluation mechanism estimates the utility of individual classifiers using reinforcement learning. Based on the obtained utilities, new classifiers are created and less useful classifiers are deleted using evolutionary mechanisms.

Pittsburgh Classifier Systems Developed in the group of De Jong at the University of Pittsburgh [54,176,177], Pittsburgh Classifier Systems consider learning to be an *off-line optimisation* process. A Pittsburgh Classifier System resembles a Genetic Algorithm that evolves a population of individuals where each individual corresponds to a set of classifiers. In each iteration, the performance of the classifier sets is evaluated and used to guide the evolutionary search.

Here, the focus is on Michigan Classifier Systems due to the strong focus on on-line learning in this thesis. Section 3.2.1 discusses XCS – a widely

used Michigan Classifier System – in some detail. Section 3.2.2 presents LCS applications for traffic signal control. Finally, the advantages and drawbacks of classifier systems are summarised in Section 3.2.3.

3.2.1 XCS

The accuracy-based classifier system XCS [213, 214] represents the state-of-the-art regarding Michigan Classifier Systems. Introduced in 1995 by Wilson, it keeps the basic ideas of Holland's original framework, but simplifies it for increased understandability and performance. This section gives a brief introduction to XCS starting with a description of its *knowledge representation* and the *performance component* (which defines the short-term behaviour of the system). In the further course of the discussion, the *reinforcement component* (which distributes received rewards among the classifiers) and the *rule discovery component* (which improves the current knowledge by applying evolutionary operators to the classifiers) are presented.

Performance component and knowledge representation

As Michigan-style system, XCS acquires knowledge by interacting with an unknown *environment*. The system receives an environmental *input*, performs an *action* based on its current knowledge, and receives a *reward* in response. The environment can, e. g., be a maze where an XCS-controlled robot needs to find a target position. In this scenario, the input codes the robot's current sensor readings, while the action corresponds to a direction of movement that (hopefully) navigates the robot closer to its target where (when reached) a reward is distributed. However, the environment might also be a function (like a binary multiplexer) where the input corresponds to a function argument and a correctly determined function value is rewarded.

The performance component of XCS determines how actions are selected based on the currently acquired knowledge on the environment. Before discussing the action selection process in more detail, the knowledge representation needs to be clarified. In XCS, the acquired knowledge is stored in a population of classifiers. Each classifier consists of five main components:

Condition C The condition defines the inputs (i. e., the part of the problem domain) to which a classifier is applicable. Initially, inputs have been limited to binary strings and conditions were defined over the alphabet $\{0, 1, \#\}$ where "#" represents a *don't care symbol* that matches both zeroes and ones. Later, Wilson and other researchers extended XCS to handle continuous-valued inputs [52, 183, 215]. Here, inputs can be vectors of real numbers and a classifier condition is a conjunctive term over several intervals.

Action A An action specifies the classifier's proposed reaction to an input. Actions are typically encoded by a set of symbols (e. g., binary strings or labels), but approaches to handle real-valued actions have been proposed recently [218].

Prediction ρ Assuming that a classifier's condition is satisfied and its action is executed, its prediction estimates the average future payoff of the executed action. The *payoff* does not refer solely to the immediate reward received from the environment, but can include the payoff prediction of the best possible action in the next iteration. The prediction is encoded by a real number in XCS, but recent extensions compute the prediction by a parameterised function (XCS-F, [119, 217]).

Prediction error ε The prediction error estimates the mean absolute deviation of a classifier's prediction with respect to the actual payoff.

Fitness F A classifier's fitness is based on an inverse function of its prediction error. It provides an estimation on the relative accuracy of the classifier with respect to other classifiers with overlapping conditions.

Besides these main components, each classifier maintains several additional bookkeeping parameters that are not discussed here in detail. Further information is available, e. g., in [37, 213].

Based on the knowledge representation, the action selection of XCS can be described. The process is depicted in Figure 3.3. Given an input, XCS determines those classifiers with a matching condition in its population

and stores them in the *match set*. The match set typically contains classifiers that advocate different actions. Since only one action can be executed in the environment in response to the received input, a selection has to take place. The action selection can be based purely on predicted payoffs (an approach followed by ZCS [212], a predecessor of XCS) or it can additionally consider the error measures that are part of the classifiers. XCS uses a fitness-weighted prediction average of all classifiers advocating the same action. The obtained prediction values for all actions are stored in a *prediction array*. To exploit the knowledge stored in the classifiers, the best action can be chosen deterministically (*exploitation*), but actions can also be selected randomly when the environment should be explored (*exploration*). In either case, all classifiers in the match set that advocate the selected action are stored in the *action set* and the selected action is performed. Based on the success of the action, the classifier system receives a reward that will be used to update the predictions and error measurements of all classifiers in the action set. The updates are performed by the system's reinforcement component.

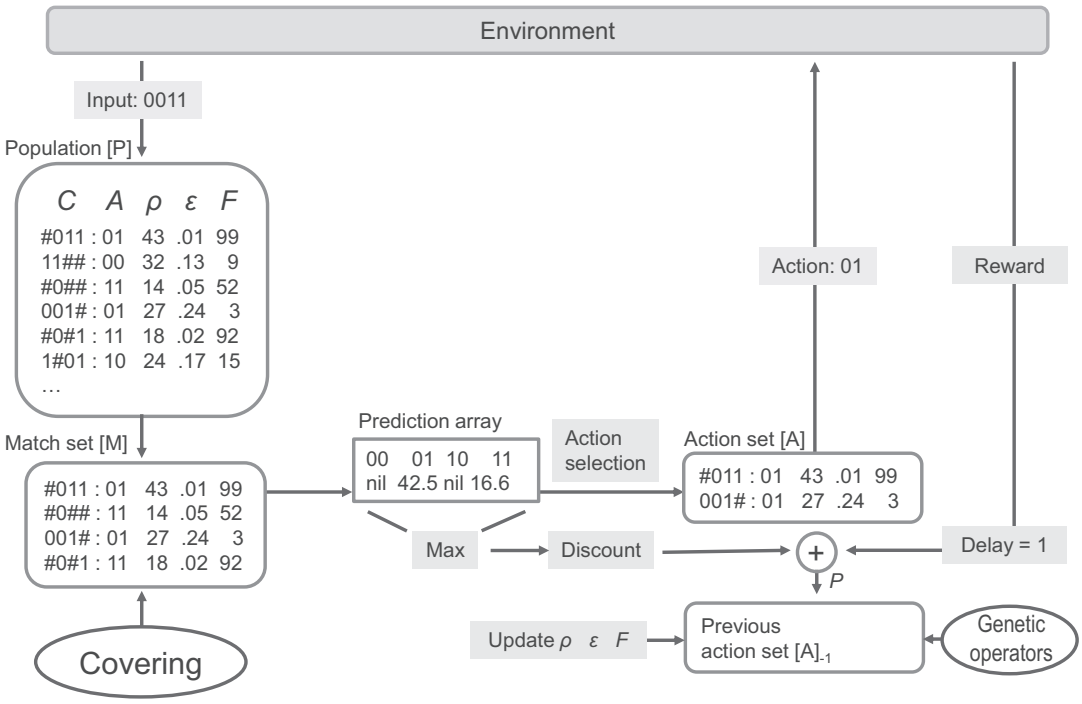


Figure 3.3: Schematic illustration of XCS (based on [214])

Reinforcement component

The reinforcement component of XCS updates prediction error, prediction, and fitness of classifiers in the action set based on the incoming reward. For the update process, single-step and multi-step problems need to be distinguished:

Single-step problems In single-step problems, every action is immediately rewarded after its execution in the environment. For every learning iteration, inputs are independent of the previous iteration's input and action.

Multi-step problems In multi-step problems, an external reward is not necessarily distributed for every learning iteration. Additionally, dependencies among successive problem instances can exist in the sense that an input may depend on previous inputs and previously executed actions. Therefore, *reward propagation* is necessary in multi-step problems.

Typical single-step environments are classification problems, where inputs have to be assigned to predefined classes. An example are Boolean multiplexer functions that have been investigated by several authors [39, 213, 214]. A Boolean multiplexer function is defined on binary strings of length $l = k + 2^k$. The first k bits of the input serve as an address that indexes one of the remaining 2^k bits that is returned as function value. The classifier system has to classify l-bit inputs depending on their multiplexer function value. This task constitutes a single-step problem, since inputs do not depend on the classification of previous inputs and the reward for a correct classification is immediately available.

Typical multi-step problems are mazes like those investigated by Wilson [213, 214]. Here, an artificial animal – called *animat* – needs to be directed to a food source in the maze. XCS receives the animat's sensor input and has to decide on the animat's movement for the next step. When the food source is reached, a reward is distributed. In the maze, the sensory input depends on the previous position (the previous input) and the previously performed movement (the previous action) of the animat. Furthermore, an obtained reward needs to be distributed among all actions that finally led to the food.

Depending on the type of problem at hand, classifier parameters are updated differently. For multi-step problems, where a reward propagation over several learning iterations is necessary, the update process is illustrated in Figure 3.3. The prediction, prediction error, and fitness values of classifiers in the action set $[A]_{-1}$ of the previous iteration are updated based on a payoff P. The payoff is calculated as sum of the reward received in the previous time step and an estimated discounted future reward which is determined as the maximum of the current prediction array multiplied by a discount factor.

Using the payoff P, the prediction error ε_j and the prediction ρ_j of each classifier j in the previous action set $[A]_{-1}$ are updated using the equations

$$\varepsilon_j := \varepsilon_j + \beta(|P - \rho_j| - \varepsilon_j) \tag{3.1}$$

and

$$\rho_j := \rho_j + \beta(P - \rho_j). \tag{3.2}$$

The variable $\beta \in [0, 1]$ is a *learning rate*. Higher learning rates lead to a faster adaptation and reduced history dependence, but result in a higher variance for fluctuating rewards.

The fitness F_j of each classifier $j \in [A]_{-1}$ is derived from its prediction error ε_j in several steps. At first, the *accuracy* κ_j is determined using a function similar to the one depicted in Figure 3.4. Classifiers with a prediction error below a threshold ε_0 are considered maximally accurate and are assigned the highest accuracy $\kappa = 1$. Less accurate classifiers with a higher prediction error receive an accuracy $\kappa < 1$ that is determined by a parameterisable power function. Using the accuracy of classifier j, its relative accuracy κ'_j with respect to the other classifiers in $[A]_{-1}$ is calculated as

$$\kappa'_j := \frac{\kappa_j}{\sum_{i \in [A]_{-1}} \kappa_i}$$

and the fitness F_j is updated to

$$F_j := F_j + \beta(\kappa'_j - F_j). \tag{3.3}$$

Besides prediction, prediction error, and fitness, the update of all classifiers in the action set also considers several additional parameters not discussed here, for details the reader is referred to [37, 213].

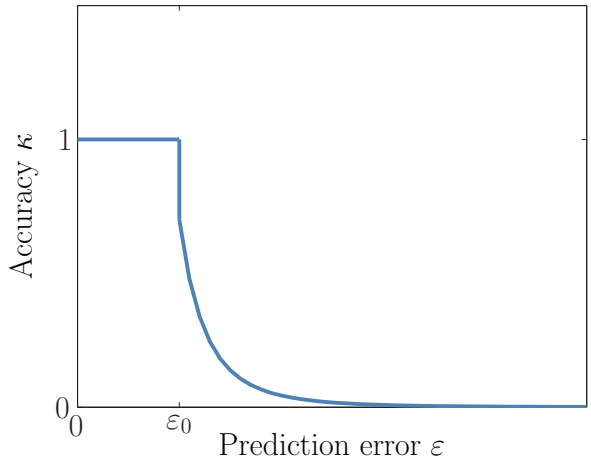


Figure 3.4: Derivation of accuracy κ from the prediction error ε

For single-step problems, updates are performed analogously to the multi-step case using Equations 3.1 to 3.3. However, since no reward propagation is necessary, the payoff P is simply determined as the current reward and the update is performed on the current action set $[A]$.

The reinforcement component updates the prediction part of existing classifiers based on received rewards, but it does neither create nor remove classifiers from the population. This is the responsibility of the discovery component.

Discovery component

In the discovery component, new classifiers are created on two occasions:

- When an empty match set is created, a *covering* mechanism creates at least one classifier with a matching condition and a randomly chosen action. The new classifier is inserted into the population and the performance component proceeds as usual.

- Genetic operators are applied to classifiers in the action set if the average time since their last genetic modification exceeds a threshold. In this case, parent classifiers are selected using proportionate [40] or tournament [38] selection considering their relative fitness with respect to other classifiers in the action set. From the selected

parents, offspring classifiers are created using crossover and mutation. After initialising the offsprings' prediction with the currently received reward and resetting the prediction error and fitness values, the offspring are included in the population.

Both mechanisms support the development of the classifier population by covering previously unknown environmental niches and by evolving new candidates from relatively fit classifiers, respectively. But besides classifier creation, the discovery component is also responsible for the deletion of classifiers. Again, two types of deletion can be distinguished in XCS:

Subsumption deletion When inserting newly created classifiers into the population or when building the action set, subsumption deletion can be applied. A classifier is subsumed by a more general classifier advising the same action if the more general classifier is sufficiently accurate (with respect to the prediction error) and experienced (with respect to the number of applications).

Population size limit To keep the population within its size limit, a classifier is selected for deletion with a probability that is proportional to the average size of the action sets it was part of. By this means, classifiers in well covered environmental niches have a higher probability to be removed. Furthermore, the experience and fitness of a classifier influence its deletion probability: If the classifier has been applied at least θ_{del} times and its fitness is lower than a δ-fraction of the population's average fitness, its deletion probability is increased.

The deletion ensures that within the population size limit only the fittest classifiers of an environmental niche are kept. With the description of the discovery component, the overall learning process is completely illustrated.

3.2.2 LCSs for traffic signal control

Several authors investigated the application of Michigan- and Pittsburgh-style classifier systems for traffic signal control.

Michigan Classifier Systems

In the area of Michigan Classifier Systems, Cao et al. investigate the use of ZCS – a predecessor of XCS – for traffic-actuated intersection control [43, 44]. In [43], ZCS controls the signalisation of a two-phased intersection controller. Based on discrete queue levels for the intersection's approaches (representing no, small, medium, or large queues), ZCS determines for how long one of the two signal phases will receive the right of way. Its actions are rewarded based on the average queue at the intersection. Later modifications of the LCS-based controller include the previous signalisation of neighbouring intersections in the input, while the actions are simplified to the selection of the active signal for the next period [44].

Results obtained from microscopic simulations show that LCSs like ZCS are in principle applicable to create traffic-actuated intersection controllers. Over a learning period, the resulting queue lengths at the controlled intersections [43] or the "traffic speeds" in the network [44] could be improved compared to signal controllers randomly switching the signalisation.

Recent works of Bull et al. investigate the use of XCS for traffic-actuated signal control [34,174]. As in the works of Cao et al., the task of XCS is to control the phase durations at a two-phased intersection. It receives the (binary coded) maximal queue length associated with both signal phases as input and decides on a possible extension of the active phase. The extension is either a binary decision whether to keep or end the active phase or alternatively the selection among four predetermined extension intervals. The reward provided to the XCS is again based on the queue length at the end of the signal phase.

The resulting XCS-based signal controller has been evaluated in microscopic simulations. Using the vehicular delay at the controlled intersections as criterion, a comparison was made to fixed-time and traffic-actuated controls for different constant and varying traffic demands. The XCS-based controller is reported to show a competitive performance compared to the reference controls.

Pittsburgh Classifier Systems

Enee and Escazut investigate the use of a Pittsburgh Classifier System to evolve control rules for a two-phased signal controller [64]. In contrast to the Michigan-style systems discussed before, Pittsburgh Classifier Systems resemble a Genetic Algorithm that evolves a population of classifier sets. In [64], each classifier set can be used for intersection control. It contains classifiers that consist of a condition specifying the state of two traffic detectors for each of the intersection's approaches and the current state of the signal. A classifier's action either switches the current signal state or leaves signalisation unchanged for the next time period.

In every generation evolved by the classifier system, all classifier sets in the population are evaluated based on the vehicular delay resulting from the repeated application of their classifiers at the intersection. Afterwards, relatively fit sets are selected and reproduced using crossover and mutation to form the next population of classifier sets.

Enee and Escazut do not compare the quality of the evolved classifier sets to reference controllers, but investigate the influence of elitism and distributed elitism on the evolutionary process. *Elitism* guarantees that the m best classifier sets of a generation are included in the next generation, while the m worst sets are discarded. In their experiments, elitism resulted in faster convergence of the population. *Distributed elitism* corresponds to the exchange of good classifier sets among the intersections of the traffic network. Distributed elitism is reported to result in convergence at a decreased amount of delay. Best results have been obtained for a combination of elitism and distributed elitism.

Criticism

Although the reported results for LCS-based signal controllers are promising, some criticism is due: Pittsburgh Classifier Systems basically resemble a Genetic Algorithm that evolves a population of classifier sets. As such, they are not well-suited for on-line applications due to the large number of required evaluations. In every new generation, each classifier set in the population needs to be evaluated many times to get a reasonable quality estimate for the combined performance of its numerous classifiers.

On-line applications are, however, advisable in traffic signal control due to the dynamic nature of traffic.

Michigan Classifier Systems are designed as on-line reinforcement learning systems that acquire their knowledge by the repeated application of classifiers in their environment. In case of traffic-actuated intersection control, this means that bad control actions are performed at the intersection for several times, before their bad quality is recognised. This has a detrimental influence on the controller performance. The problem is usually handled by introducing a learning phase for exploration that precedes the evaluation period during which the previously acquired knowledge is exploited.

The two-levelled learning approach presented in Chapter 5 simultaneously performs exploration and exploitation by using a simulation environment that runs in parallel to the controlled environment. The simulation is used to explore new classifiers, while the previously learnt knowledge is exploited in the real environment. The two-levelled learning architecture therefore allows to widely eliminate the detrimental effects of mistakes unavoidable in the learning process.

A further objection regarding the LCS-based approaches previously discussed is related to the missing coordination mechanisms: None of the presented approaches explicitly coordinates neighbouring intersections to form progressive signal systems, although the coordination of signals can be beneficial especially in urban areas (see Chapters 7 and 8). All approaches rely on a loose, indirect coupling of neighbouring intersections by detecting approaching vehicles [64] or existing queues [34, 43, 174]. Only in [44], the previous signalisation of neighbouring intersections is included in the LCS input. Whether these loose couplings help to improve the signalisation or lead to the traffic-responsive formation of progressive signal systems is, however, not reported.

A commonality of the discussed LCS-based approaches to signal control is the simplicity of their scenarios. All mentioned applications implement two-phased controllers and, therefore, the simplest possible phase system. The classifier conditions are of a limited complexity, considering traffic indications for maximally four one-laned approaches. At realistic intersections, traffic measures need to be handled for several signal groups and complex phase systems are common. This results in more complex

conditions and actions for the classifier rules. Unfortunately, classifier systems are known to need an increasing number of learning iterations when the conditions and actions of their rules get more complex [159].

3.2.3 Summary

What should be remembered regarding LCSs is that they are rule-based reinforcement learning systems that can be distinguished in Pittsburgh [54, 176, 177] and Michigan Classifier Systems [95, 97]: Pittsburgh-style systems are closely related to Genetic Algorithms and are especially suitable for off-line optimisation, while Michigan-style systems are on-line learning systems. A widely used Michigan-style system is Wilson's XCS [213, 214] which aims at learning classifiers that are on the one hand accurate in their prediction and that are on the other hand maximally general with respect to their condition.

An advantage of LCSs is that they are well-suited for many machine learning tasks since their only requirement is a scalar reinforcement feedback provided by their environment. As a result, the LCS framework is widely applied in a large area of real-world applications in domains such as data mining, modelling, optimisation, and control [33]. Classifiers learnt by an LCS are easy to interpret and to understand since they are available in a human-readable format. Therefore, the classifier population provides a plausible, understandable model of the learnt concepts.

Drawbacks of LCSs are the number of environmental iterations necessary during the learning process. Especially when classifier conditions are complex and the number of available actions of the LCS are large, learning can be a slow and time-consuming process [159]. This might be an explanation for the limited complexity of LCS-based intersection controllers previously investigated in the literature. Another drawback is that the quality of a classifier is determined by its execution in the real environment. As a result, bad control actions have to be executed several times, before they can be recognised and discarded. Since this has a detrimental influence on the LCS performance, learning periods for exploration precede the productive application of LCSs in many cases. Both issues are addressed in Chapter 5 by the introduction of a two-levelled learning architecture that combines the LCS with a simulation-based optimisation component.

Regarding LCS literature and research, Lanzi's overview [118] gives a good account on the historical development of the field, while a good textbook on (Michigan-syle) LCSs has been published by Butz [37]. To follow recent developments, the yearly *International Workshop on Learning Classifier Systems* held in conjunction with the *Genetic and Evolutionary Computation Conference* (GECCO) can be recommended. Furthermore, a bibliography on LCSs is maintained by Kovacs[4].

In the following, attention is given to self-organising traffic signal systems that consist of locally interacting intersections which obtain a coordination without centralised control. The intersections might learn in their local environment or can follow a predefined logic.

3.3 Self-organisation in traffic control

In recent years, self-organisation has received growing attention in the traffic engineering community. Self-organising signal systems try to achieve a quick network-wide adaptation to changing traffic demands that results from local interactions of neighbouring intersections. The approaches can be distinguished by the communication requirements of their intersection controllers: *Communicating intersections* rely on a dedicated communication link to exchange information on traffic demands or other data relevant for the traffic-responsive signal coordination with their neighbours. *Non-communicating intersections* achieve a coupling with their neighbours by an early detection of arriving vehicles which typically requires more sophisticated detection equipment. Examples for both system types are discussed in Sections 3.3.1 and 3.3.2, respectively. The discussion is concluded with a summary in Section 3.3.3.

3.3.1 Non-communicating intersections

As examples for self-organising traffic control systems that do not rely on a communication link among neighbouring intersections, the approaches suggested by Gershenson et al. [48,80,81] and Lämmer et al. [114,116,117] will be discussed in the following.

[4] T. Kovacs. A Learning Classifier Systems Bibliography: `http://www.cs.bris.ac.uk/~kovacs/lcs/search.html`

Platoon-based self-organisation

The self-organising traffic light (SOTL) system proposed by Gershenson et al. [48, 80, 81] relies on traffic-actuated intersection controls that are operated using relatively simple local rules. Three different SOTL variants with increasing complexity have been proposed for two-phased intersections:

SOTL-request In the simplest variant, vehicles waiting at or approaching a red light are counted. In every time step, the counts are summed up and whenever the sum reaches a predefined threshold, the signalisation changes and the sum is set to zero. Using this SOTL-request rule, long vehicle queues or large platoons approaching an intersection can quickly obtain a green signal, while small vehicle groups have to wait for a longer period of time. However, at high traffic densities, the rule leads to frequent changes in the signalisation that result in increased lost times.

SOTL-phase To avoid frequent signal changes, SOTL-phase extends SOTL-request with a constraint on the minimal phase duration: The switching condition is only evaluated if the current phase has been active for a certain period of time.

SOTL-platoon Finally, SOTL-platoon extends SOTL-phase with a mechanism regulating the size of vehicle platoons: If there are vehicles close to the stop line of a currently green signal, the signalisation will not be changed unless the number of approaching vehicles is above a threshold. The additional rule ensures that the tail of a platoon is not separated from its head unless the tail is large enough to form a platoon on its own.

Similar to the traffic-actuated controls discussed in Section 2.3, the self-organising traffic lights adapt their signalisation by taking into account current vehicle queues and gaps in the approaching traffic (which are implicitly used to detect separate platoons). However, the difference is in the way neighbouring intersections are coordinated. While conventional controls rely on a predefined coordination that requires a fixed cycle length at each intersection, the self-organising traffic lights work *acyclic*,

i. e., without a fixed cycle length. A coupling of neighbouring intersections is achieved by the early detection of approaching vehicle platoons that can be incorporated in the signalisation before their arrival at the intersection due to the flexibility of the acyclic signalisation.

A comparison of the different SOTL approaches is available in [80, 81]. Furthermore, the SOTL-platoon approach has been evaluated for a four-laned one-way avenue of two-phased intersections at Brussels, Belgium [48,81]. The simulation study considered the avenue's daily traffic demand and compared SOTL-platoon to a fixed coordination used in the field. Reductions of about 50 % are reported with respect to the resulting average waiting times. While the self-organising approach is flexible and adapts the local signalisation based on simple rules, its drawback is in the substantial detection equipment necessary for the early detection of arriving platoons.

Pressure-based self-organisation

Lämmer et al. [114, 116, 117] propose a decentralised control principle for traffic lights that shares the general ideas of Gershenson. The signalisation at an intersection is adapted based on "pressures" generated by approaching vehicles. No predefined cycle times or phase sequences are used: At every intersection, the signal groups are dynamically combined in response to local traffic demands considering safety-critical constraints (like minimum green times or clearing times), only. As a result, local optimisations can be performed on a relatively unconstrained search space.

As before, a coordination of neighbouring intersections is reported to emerge from the local optimisations: Since stopped vehicle platoons would increase the local delay, the local optimiser avoids stopping platoons whenever this is reasonable. Therefore, a dynamic, traffic-responsive coordination of signals results from the local optimisations.

The three main components locally controlling the intersections are briefly sketched in the following:

Anticipation The anticipation component estimates additional delays occurring if a signal group will receive the right of way at a later point in time. The calculation does not only consider already

queued vehicles, but also approaching vehicles that will have to wait until the existing queues are cleared.

Optimisation Based on the results of the anticipation component, a combination of non-conflicting turning movements that results in the lowest delay at the intersection is selected by the optimisation component. In the process, a dynamic priority is assigned to each turning and a selection of turnings with highest priorities (or *pressures*) receives the right of way.

Stabilisation Finally, the stabilisation component ensures that minor turnings with relatively small flows do not exceed a maximal red time. Without stabilisation, the optimisation might discriminate minor turnings against larger traffic movements, since the savings obtainable for minor turnings are small. Details on instabilities are available in [115].

The self-organising approach has been evaluated in a simulation study for a subnetwork of 13 intersections surrounding the train station "Dresden Mitte" at Dresden, Germany [117]. The investigated subnetwork is operated using three predefined coordination programmes with local traffic-actuated control provided by VS-Plus. In the study – that was conducted using the microscopic traffic simulator VISSIM [66] – these coordinations served as reference solution. Using the self-organising approach, a reduction of 9 % with respect to the vehicular delay could be obtained. For public transport vehicles – that execute a higher pressure on a signal – the delay could be shortened by 56 %. The average red time for pedestrians and cyclists could be reduced by 36 %, since they are served in parallel to vehicular movements whenever possible.

Judging the results of the simulation study, the self-organised emergence of coordinations within the network works well. However, drawbacks of Lämmer's approach are in its detection requirements. The study assumed an "ideal vehicle detection", i. e., the positions of all vehicles are known in the simulation. Further tests regarding more realistic scenarios with incomplete detection are necessary. Furthermore, the dynamic composition of turning movements might lead to acceptance problems for road users since the system behaviour cannot be easily understood from their point of view.

3.3.2 Communicating intersections

A game-based mechanism proposed by Bazzan [13] serves as example for a distributed approach to traffic signal coordination that relies on communicating intersections. In the approach, intersections are modeled as individually-motivated agents. Each agent possesses a set of predefined signal plans to choose from. The selection process is based on local traffic flows at the intersections as well as on the results of *coordination games* that are played among neighbouring agents. In detail, individual state-change periods, payoff-getting periods, and learning periods determine the signal plan selection:

Individual state-change periods When a change of the local traffic flows is detected at an intersection, an individual state-change period takes place. In this case, the selection probabilities of the predefined signal plans are determined considering local traffic flows only.

Payoff-getting periods During payoff-getting periods, the intersection agents play two-player coordination games with their neighbours in order to achieve a coordination that is beneficial for the network. The combination of signal plans selected by the agents is rewarded by a payoff which is determined from a payoff matrix that has to be explicitly formalised by the designer of the system.

Learning periods During learning periods, the selection probabilities of signal plans are adapted based on the previously received payoffs. Giving a greater significance to recently received payoffs, a fitness value is calculated for each signal plan. The selection probabilities of the signal plans are then updated to represent their relative fitness.

The principal applicability of Bazzan's approach is demonstrated in a simple scenario of an arterial road consisting of ten intersections. Each intersection agent has to choose between two signal plans, each of which favours one of the two arterial directions over the other. The distributed approach is compared to a centralised controller that coordinates all traffic lights in one of the arterial directions. While the centralised coordination reduces traffic jams at the arterial when the vehicle flow is

clearly higher in one direction, the agent-based approach proves to be better in situations where the flow in both directions is nearly equal. In these situations, it is advantageous that the intersection agents can break with the coordination in order to cope with their local traffic demands when this becomes necessary.

In contrast to the approaches discussed in Section 3.3.1, intersection agents in Bazzan's approach need communication capabilities to participate in the coordination games. The payoff matrices required in these two-player games have been identified as shortcoming of the approach [14], since they have have to be explicitly formalised by the system designer. Their definition requires knowledge on the quality of different coordination options under different traffic demands. The definition process is therefore time-consuming and not well-suited for on-line applications. Another weakness of the approach is in the restriction to predefined signal plans that limit the flexibility of Bazzan's approach in on-line applications. While the decentralised traffic control system proposed in Chapter 7 also relies on communicating intersections, it does not require predefined signal plans or prior knowledge on coordination options.

3.3.3 Summary

Self-organising traffic signal systems are characterised by a quick adaptation to changing traffic demands that results from a decentralised interaction of the network's intersections. Their traffic-responsive coordination emerges from local decision processes taking place at each intersection, without a centralised authority being involved. The coordination can either rely on the early detection of approaching vehicles (which requires sophisticated detection equipment) or on an additional communication link for the transmission of traffic or signal plan data. Some approaches (like those in Section 3.3.1) depend on relatively simple, predefined local decision rules, while others (like the one discussed in Section 3.3.2) additionally make use of learning mechanisms.

The proposed self-organising approaches are too numerous to be comprehensively discussed in this section. Self-organisation is not only investigated to improve traffic signals, but also to replace them. The anticipated reservation-based mechanisms treat a signalised intersection

as multi-agent system and assume that driver agents request an intersection agent to reserve time-space slots required to safely cross the intersection [61, 167, 200].

To follow recent developments, the workshops on *Agents in Traffic and Transportation* at the yearly *Conference on Autonomous Agents and Multiagent Systems* (AAMAS) can be recommended. Furthermore, collections on agents in traffic and transportation [15, 106] and a recent survey [14] are at hand. However, the available works have been conducted to investigate differing research questions and exhibit a varying degree of sophistication. Some do not strive to create a traffic control system in the first place, but use the traffic scenario solely as distributed test case for, e. g., reinforcement learning [49]. Other works rely on simplifying assumptions that render the proposed solutions uninteresting for real-world applications. Therefore, articles on self-organising traffic systems should be reviewed with a critical attitude.

CHAPTER 4

Organic Computing and the observer/controller paradigm

While the previous chapters presented the state of the art in traffic signal control and discussed learning and optimisation approaches, this chapter is dedicated to Organic Computing and its observer/controller paradigm. Organic Computing is a research field that investigates how to design adaptive and self-organising technical systems, its vision is in the focus of Section 4.1. The observer/controller paradigm provides a generic design framework for organic systems. Since the framework forms the basis for the adaptive learning intersection controller developed as part of this thesis, it is introduced in Section 4.2. Section 4.3 briefly reviews related frameworks, before the discussion is concluded in Section 4.4.

4.1 Organic Computing

The advancement of technology leads to increasingly powerful technical systems that provide their users with functionalities and services that

were unthinkable only a few years ago. An example are modern cars that are endued with an increasing number of electronic components. Numerous processors and embedded systems provide motor control (e. g., injection and ignition), safety (e. g., air bags), driver assistance (e. g., proximity cruise control), and comfort functions (e. g., parking assistance). Often, several embedded systems are interconnected and form a complex communication network, e. g., when a system for tyre pressure monitoring relies on ABS sensor data.

While the advancement of technology provides increased functionality, comfort, and safety to the user of a technical system, it poses tremendous challenges to the system designer. For him, it becomes more and more difficult to foresee all possible configurations and to explicitly specify the entire behaviour of a complex system on a detailed level at design time. In particular, when a system consists of many interacting components, it may exhibit new, emergent properties that are difficult to anticipate.

The increasing complexity of technical systems calls for new system architectures, since issues not anticipated in the design stage of a system have to be dealt with at run-time. This requires *adaptive systems* that can adjust themselves to changing conditions and *self-organising systems* that consist of locally interacting components which obtain an emerging global behaviour without centralised control [63, 81]. Regarding the necessity of such systems, Schmeck [168] states that

> "It is not the question whether self-organized and adaptive systems will arise but how they will be designed."

A research field that focuses on adaptive and self-organising systems is *Organic Computing* (OC) [137, 139, 168, 201]. OC aims to design technical systems that are *trustworthy*, *robust* (with respect to disturbances or failures), and *flexible* (with respect to externally provided goals). Additionally, organic systems shall exhibit *self-x-properties* like self-configuration or self-optimisation.

There are, however, challenges in the design of organic systems: Adaptation requires the capability to monitor a system and its environment in order to influence its behaviour with respect to externally provided goals. The process of efficient monitoring and control may demand for machine learning and optimisation techniques, so an adaptive system

can learn adequate responses to conditions unforeseen by its designer. Unfortunately, learning involves unsuccessful experiments that cannot be conducted in the productive environment, so learning in on-line scenarios poses a first challenge to the design of organic systems.

A second challenge imposed by self-organisation are emergent properties that arise in complex systems of interacting components where the study of individual components reveals little about the system-wide behaviour. Emergent properties can be desired or undesired, but they make the behaviour of a self-organising system difficult to predict. Therefore, it may become necessary to observe and temporarily control a self-organising system in order to suppress unwanted emergent effects and achieve *controlled self-organisation*. In OC, both challenges are addressed by the generic observer/controller architecture.

4.2 Generic observer/controller architecture

In order to assess the behaviour of a technical *System under Observation and Control* (SuOC) and – if necessary – for a regulatory feedback to control its dynamics, it is assumed that a *generic observer/controller architecture* is required. The general idea of an observer/controller has been first proposed in [136] and was later refined [23, 157, 159]. The main objectives of the architecture are

- to reconfigure the SuOC to achieve an optimal system performance despite dynamically changing environmental conditions and
- to achieve controlled self-organisation by handling emergent effects resulting from the interaction of the SuOC's components.

To perform these tasks, the observer/controller has a set of sensors and actuators to measure system variables and influence the SuOC. As depicted in Figure 4.1, the *observer* collects data from the SuOC and computes indicators characterising its global state and dynamics. This process includes the preprocessing of monitored data, an analysis to derive system-wide indicators, and a prediction of future developments. The results of these steps are aggregated to *situation parameters* characterising the observed or future system state.

The situation parameters are evaluated by the *controller* with respect to a user-defined goal. The controller decides whether an intervention is required and, if so, what action would be most appropriate. This decision is based on a mapping of situation parameters to appropriate actions that is continuously evaluated with respect to its performance (Level 1). Based on results of the evaluation and possibly on estimations obtained from a simulation model, this mapping can be adapted, refined, or extended (Level 2), making the controller a learning component.

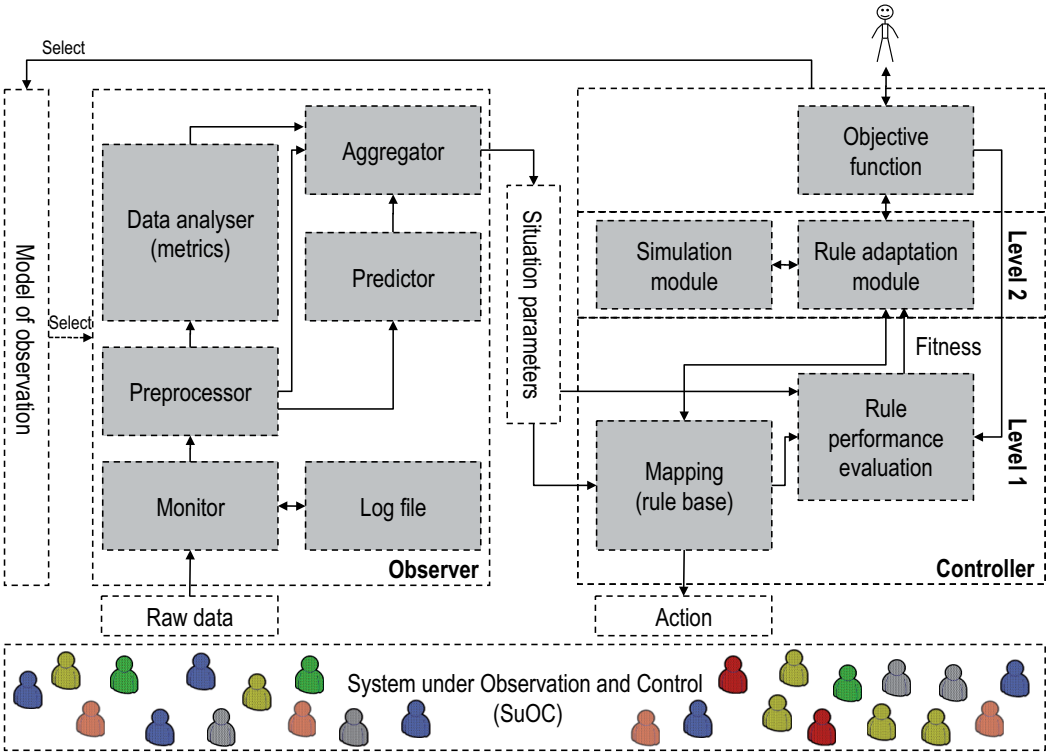


Figure 4.1: Generic observer/controller architecture (based on [159])

The combination of SuOC and observer/controller forms an organic system and will be discussed in more detail in the following. Section 4.2.1 focuses on the SuOC, while Sections 4.2.2 and 4.2.3 are dedicated to the observer and controller, respectively.

4.2.1 System under Observation and Control

The SuOC can be an arbitrary technical system that benefits from supervision and adaptation to changing conditions. Examples include classically engineered systems like off-highway machines where different working cycles can be detected to reduce the machine's fuel consumption [221], elevator groups where bunching effects are prevented [155], or – as in this thesis – traffic lights that can be adapted to changing traffic demands.

The SuOC may also consist of a large collection of autonomous objects that interact in their environment. In this case, the observer/controller can detect emergent phenomena resulting from the local interactions and intervene in the SuOC in order to suppress or support (un)desired emergent effects. Examples include service-oriented architectures that can be combined with observer/controller components to cope with the increasing complexity of IT systems [122]. The variety of applications underlines the generic character of the observer/controller framework that will now be discussed in more detail.

4.2.2 Observer

It is the observer's task to measure and quantify the current state of the SuOC and to predict its future development. The observation process consists of the steps monitoring, preprocessing, data analysis, prediction, and aggregation. An observation model customises the observer by selecting the observable attributes of the SuOC and by determining appropriate analysis and prediction methods.

Monitor and log file The monitor samples attributes of the SuOC according to a given sampling rate. The observed system data consists of individual data on the level of single elements (like a traffic detector's vehicle count) or some global system attributes (like weather conditions). All measured data is stored in a log file for every loop of observing/controlling the SuOC. The stored time series form the basis for the preprocessing, analysis, and prediction steps.

Preprocessor Typical tasks performed during preprocessing include the smoothing and filtering of stored time series (e. g., from a traffic de-

tector) and the extraction of derived attributes (like the calculation of queue lengths from vehicle counts at two detector locations). The preprocessed data is then used in the data analysis and prediction steps.

Data analyser The data analyser provides a system-wide description of the SuOC's current state. The implemented analysis techniques largely depend on the observed system and the purpose of the observer/controller. In the context of intelligent traffic systems, vehicular delays and stops might be estimated using a traffic model.

Predictor While the data analyser is dedicated to the current system state, the predictor's task is to forecast future developments. This enables the controller to base its control decisions not only on historic data, but also on predicted developments. Prediction techniques are again specific to the organic system's domain. For intelligent traffic systems, an overview of short-term forecasting methods is provided in [202].

Aggregator The results of the preprocessor, data analyser, and predictor are handed on to the aggregator where they are combined to situation parameters which are transmitted to the controller.

Depending on the requirements of the controller, the obtained situation parameters can be adapted using the *model of observation*. The model specifies which properties of the SuOC are observed and which sampling rate is used for observation (this selection is obviously limited by the available detectors and their capabilities). Furthermore, the applied analysis and prediction methods are selected. By influencing the model of observation, the controller can focus in detail on critical aspects within the SuOC (like an incident blocking a road and thereby influencing the traffic flow) even when its resources (like communication bandwidth, processing power, or energy) are limited.

4.2.3 Controller

Based on the received situation parameters, the controller will influence the SuOC to achieve the goals given by the user. The controller is inter-

nally composed of two levels: *Level 1* is dedicated to the *on-line learning* of appropriate actions for received situation parameters. It consists of a mapping component that assigns possible actions to known situations. Furthermore, the performance of stored situation-action mappings that have been applied in the SuOC is evaluated by a performance evaluation component.

Level 2 aims at improving the mapping of Level 1 by providing *off-line optimisation* capabilities. It is composed of an adaptation module and a simulation model. Using optimisation or machine learning approaches, the adaptation module creates additional mappings and can rely on the simulation model in the process. The main components of both layers are discussed in the following:

Mapping The mapping component is responsible for an immediate reaction to received situation parameters. It stores previously learnt situation-action mappings (e. g., as LCS rule base) which determine the reaction to known situations. When no matching mappings are available for an observed situation, an action might be deduced from stored mappings for similar situations. This reaction can be suboptimal, though. Therefore, an additional mapping might be requested from the adaptation module.

Performance evaluation The performance evaluation component calculates quality updates for the situation-action mappings based on their performance observed in the SuOC. When the execution of an action changes the state of the SuOC, this change is subsequently reflected in the situation parameters derived by the observer. The controller's performance evaluation component evaluates the change and/or the new situation with respect to the user-specified objective function and updates previously applied situation-action mappings through the adaptation module. Machine learning techniques like reinforcement learning [186] can be used here. The update of existing mappings resembles an on-line learning based on observations in the SuOC.

Adaptation module The adaptation module's main task is to create new situation-action mappings and to delete mappings of insufficient

quality. For the exploration of additional mappings, several optimisation or machine learning approaches can be used which are supplemented by a simulation model that allows for the safe evaluation of candidate solutions. The adaptation module resembles an off-line optimisation that is performed in parallel to the on-line learning mentioned above.

Simulation module The simulation module supports the exploration of new situation-action mappings. It allows to safely estimate the effect of a new action or mapping before its application in the SuOC. Simulation-based estimations are typically much faster than evaluations in the real systems, but abstractions in the simulation model can lead to imprecise results. By calibrating the model, imprecisions can be kept small and will be quickly corrected by the performance evaluation component.

Objective function The user-defined objective function guides the learning and optimisation processes on both levels of the controller. It is used for fitness evaluation in the performance and adaptation modules.

A remarkable speciality of the controller is the two-levelled design. The combination of on-line learning (Level 1) and off-line optimisation (Level 2) exhibits several advantages compared to other control loops or learning mechanisms like those discussed in Section 4.3:

- The simulation-based off-line optimisation enables the controller to find appropriate situation-action mappings without having to test different alternatives in the SuOC. The latter could be detrimental, as testing potentially bad strategies in the SuOC can result in an unacceptable performance or might even cause it to fail permanently.

- Typically, simulation-based evaluations of situation-action mappings are significantly faster than evaluations in the SuOC. Assuming a limited amount of time for optimisation, heuristics using simulations can rely on a larger number of evaluations for training and search.

- Although optimisation on Level 2 takes time, the mapping component of Level 1 allows for a quick reaction of the controller by acting as a memory for learnt knowledge. Even when unknown situations are observed, the controller can use situation-action mappings for similar known situations to determine an immediate reaction (that might be suboptimal, though). In parallel, the optimisation on Level 2 can be started, so a specially optimised situation-action mapping will be available in the future.

- The model-based optimisation on Level 2 is subject to simplifications in the model. Therefore, the best action determined by simulations is not necessarily the best action within the SuOC. By adapting the estimated quality of the situation-action mappings, Level 1 can fine-tune the solutions from the adaptation module and will correct possible inconsistencies.

Overall, the generic observer/controller architecture provides a framework that is widely applicable to a large range of technical systems. Before an adaptive learning intersection controller based on the observer/controller architecture is presented in Chapter 5, related architectures are briefly discussed in the following.

4.3 Related architectures

Due to the importance of autonomous control in technical systems, several control loops similar to the generic observer/controller architecture have been proposed in different fields. Section 4.3.1 provides some helpful references on loops closely related to the observer/controller architecture, while Section 4.3.2 focuses on architectures for simulation-based optimisation and learning at run-time.

4.3.1 Control loops

The most relevant control loops for traffic signal control are adaptive network control systems that have already been discussed in Section 2.4.

Other related approaches include IBM's Autonomic Computing[1] and the Operator-Controller Module developed in the Collaborative Research Centre 614[2]:

Autonomic Computing IBM's Autonomic Computing initiative focuses on the construction of IT systems that manage themselves according to an administrator's goals. The system's hardware and software components are treated as *managed elements* that are equipped with an *autonomic manager* executing a *monitor-analyse-plan-execute* (MAPE) loop to achieve a self-management of the IT system [102, 178, 179].

Operator-Controller Module To realise self-optimising mechatronic systems (that combine technologies applied in mechanical and electrical engineering), an Operator-Controller Module has been proposed [36, 142]. The Operator-Controller Module has a strong focus on real-time constraints. A *controller* featuring a predefined number of alternative control strategies directly affects the mechatronic system. It is complemented by a *reflective operator* that may modify the controller and initiate switches between control strategies. A *cognitive operator* gathers information concerning the system and its environment to improve the system behaviour.

Both, the MAPE loop of Autonomic Computing and the Operator-Controller Module differ from the observer/controller architecture in a number of respects: In contrast to MAPE, the observer/controller puts a strong emphasis on the importance of (two-levelled) learning. The Operator-Controller Module, on the other hand, exhibits a two-levelled design, but does not possess the observer's distinct analysis and prediction functionality. For an extensive discussion of differences that also considers other control loops like *Model Predictive Control* [41] or *Sense-Plan-Act* approaches in robotics [31], the reader is referred to [159]. Robotic architectures that rely on simulation-based optimisation to improve their behaviour at run-time are also discussed in the following.

[1] IBM Corporation. Autonomic Computing website: `http://www.research.ibm.com/autonomic/`

[2] Universität Paderborn. Collaborative Research Centre 614 website: `http://www.sfb614.de/`

4.3.2 Simulation-based optimisation at run-time

Approaches that rely on a simulation-based optimisation at run-time have been predominantly proposed in the area of robotics [20, 92, 150]. Here, special attention will be given to an approach called *Anytime Learning* that bears several similarities with the two-levelled learning in the observer/controller architecture.

Anytime Learning [92, 150] aims at controlling an agent (e. g., an autonomous robot) in changing environments by endowing it with a continuous learning capability. As depicted in Figure 4.2, the agent possesses an execution system and a learning system: The *execution system* controls the agent's interaction with its environment based on a knowledge base containing its active strategy. The *learning system* attempts to improve the execution system by continuously testing new strategies against a simulation model.

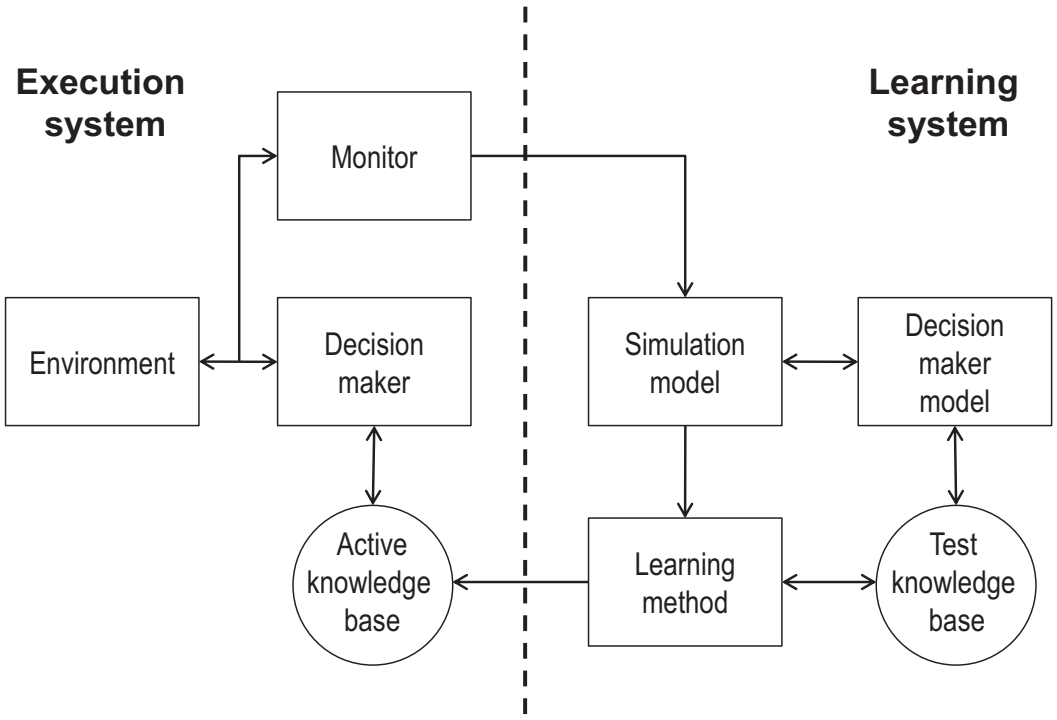


Figure 4.2: Anytime learning system (based on [92, 150])

The communication between execution and learning system is bidirectional: The learning system updates the execution system's knowledge

base when – based on the tests performed on the simulation model – a strategy is found that performs better than the active strategy. In the other direction, the execution system monitors the agent's behaviour in the environment and dynamically modifies the learning system's simulation model whenever discrepancies are detected. In this case, the learning process is restarted on the modified model. Whether a complete restart is necessary or a more graceful adaptation is possible depends on the employed learning method.

Despite the obvious similarities to the generic observer/controller architecture that also combines a knowledge base (Level 1) with a simulation-based optimisation approach (Level 2), there are important differences:

- In Anytime Learning, discrepancies in the simulation model are handled by adapting the model parameters and restarting the learning process. Such a model calibration is also possible in the observer/controller architecture (performed by the adaptation module based on data gathered by the observer), but calibration can be a tedious task (especially when performed on-line). Therefore, the observer/controller architecture additionally relies on the performance evaluation component to handle small discrepancies between simulated and real environment using machine learning techniques like reinforcement learning.

- In Anytime Learning, the knowledge base determining the controlled agent's active strategy is replaced by a new strategy from the learning system. Therefore, a new learning process is required when previous environmental conditions reoccur, leading to a reduced performance during the (repeated) learning period (see the results reported in [92]). To avoid this drawback, the observer/controller architecture stores previously learnt situation-action mappings and considers only the mappings matching the current environmental conditions in the action selection.

Despite the existence of related architectures and learning approaches in the literature, the observer/controller approach represents a new mechanism for the autonomous control of technical systems with respect to the goals of a human user. In the following section, its main features will be summarised before it is applied to traffic signal control in Chapter 5.

4.4 Summary

The increasing complexity of technical systems calls for adaptive and self-organising architectures that relieve system designers from explicitly specifying the entire behaviour of a complex system at design time. The anticipated systems should be designed with respect to human needs and have to be trustworthy, robust, and flexible. They will exhibit self-x-properties, learn about their environment over time, survive attacks and breakdowns, adapt to their users, and react sensibly even if they encounter a new situation for which they have not been explicitly programmed. Due to their life-like properties, they are called organic systems.

To create adaptive learning systems that perform well in dynamically changing environments and to achieve controlled self-organisation by handling emergent effects that are inherent in complex systems, a generic observer/controller architecture has been proposed. The architecture has a framework character and outlines the main components of an organic system: The observer collects data from a (self-organising) SuOC and computes some indicators characterising the system state and dynamics. Based on these situation parameters, the controller decides whether an intervention is required and, if so, what action would be most appropriate. The decision process involves a two-levelled learning that combines simulation-based optimisation and on-line learning.

Possible applications of the generic observer/controller architecture are manifold. They include the control of robot swarms [131], the optimisation of communication networks [189, 193], working cycle detection in off-highway machines [221], and elevator control [155] among others (see [190] for a recent survey). The works emphasise different aspects of the framework, but not every application uses all functionalities proposed for the observer/controller architecture. In the next chapter, the framework is adapted to traffic signal control at an intersection, emphasising the importance of two-levelled learning for the control task. The adaptation constitutes one of the main contributions of this thesis.

CHAPTER 5

An observer/controller architecture for traffic control

The previous chapter presented the observer/controller architecture as a generic framework for the design of organic systems. Here, the architecture is implemented to obtain an adaptive learning intersection controller. Section 5.1 provides a system overview before the observer/controller implementation is presented in detail. Section 5.2 focuses on the SuOC and discusses different types of traffic light controls. The observer, its detection requirements, and the steps necessary for data processing and analysis are discussed in Section 5.3, while Section 5.4 is dedicated to the two-levelled learning and optimisation in the controller. Finally, the real-world deployment of the organic system and its simulation-based evaluation are discussed in Section 5.5.

5.1 Overview

The focus of this chapter is on an adaptive learning intersection controller that is based on the generic observer/controller architecture. A signalised

intersection forms the SuOC that is extended by an observer/controller to obtain an adaptive learning system that allows for the on-line reconfiguration of signal plans (see Figure 5.1). The observer monitors the local traffic demand, analyses the active signal plan's performance, and predicts future developments. Based on the resulting situation parameters, the controller reconfigures the intersection's signal plan on-line. A modified LCS forms the controller's first level that is responsible for on-line signal plan selection. As reinforcement learning mechanism, the LCS keeps track of a rule base that maps traffic demands to appropriate signal plans. The controller's second level is responsible for signal plan optimisation and relies on an EA as optimisation technique. The EA evaluates the quality of candidate solutions by microscopic simulations (or, in case of fixed-time plans, by approximations) and provides the optimised signal plans to the LCS where they become available for activation in the SuOC.

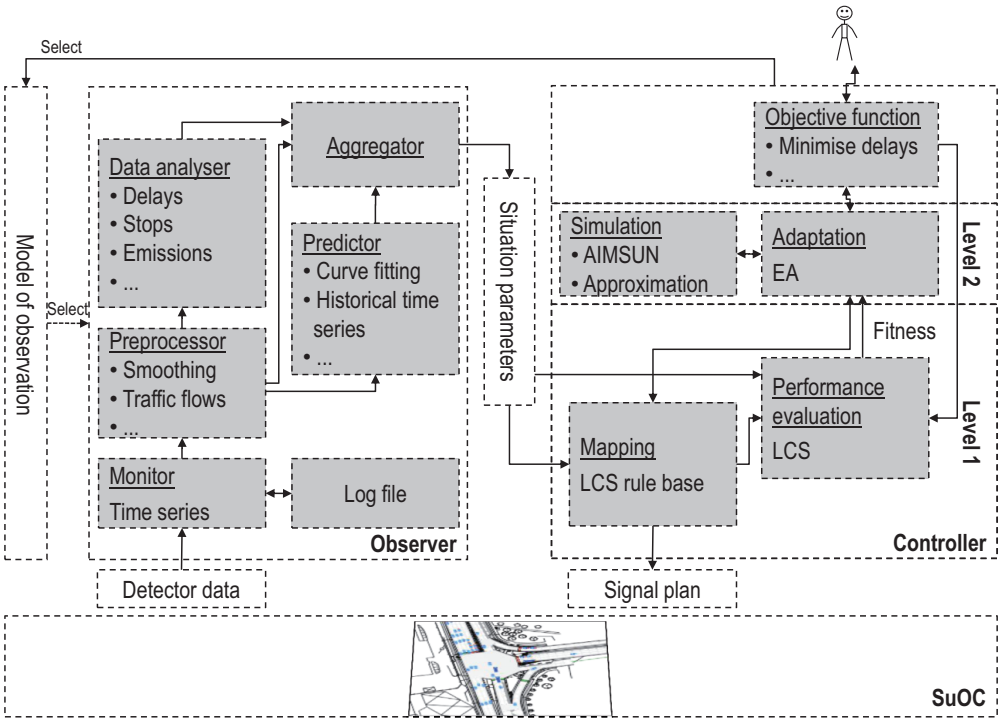


Figure 5.1: An observer/controller architecture for traffic signal control

The details of the observer/controller implementation are presented in the remainder of this chapter, starting with a discussion of the SuOC in Section 5.2.

5.2 System under Observation and Control

In the SuOC, a traffic light controller (TLC) is responsible for physically setting the intersection's traffic lights. Different TLC types may be implemented in the SuOC. Section 5.2.1 focuses on fixed-time controls, while traffic-actuated variants are discussed in Section 5.2.2. Finally, the discussion of the SuOC is concluded in Section 5.2.3.

5.2.1 Fixed-time controls

Fixed-time controls (see Section 2.2) are normally configured at design time based on historical traffic data. During their operation, the controls do not adapt their signalisation to changing demands. Therefore, a day-time dependent switching among several predefined signal plans is common practice. Since no on-line evaluation or optimisation takes place, fixed-time controls are affected by an ageing effect [17]. Long term changes in traffic lead to a decreasing quality of control and necessitate regular signal plan reviews by a traffic engineer. In practice, these reviews are costly and time-consuming.

By equipping a fixed-time controller with an additional observer/controller architecture, its signal timings can be adapted to changing traffic demands. The active signal plan is evaluated and optimised at run-time, thereby keeping a constantly high signalisation quality. In principle, all four basic signalisation parameters can be reconfigured on-line by the observer/controller:

Phase durations and cycle For a given signal plan, cycle time and phase durations can be modified considering constraints like minimum green times or maximum cycle lengths. Although cycle time and phase durations are no safety-critical parameters (as long as they respect the given constraints), their appropriate configuration has a significant influence on the vehicular delay at the intersection (see Figure 2.6). Therefore, both parameters are considered in the experiments conducted for this thesis (see Chapter 6).

Phase sequence The phase sequence of a fixed-time controller can be considered for on-line reconfiguration to minimise the clearance times

and to improve the coordination with neighbouring intersections. An observer/controller might either switch among predefined phase sequences specified at design time by a traffic engineer or might create a new phase sequence at run-time. In the second case, a predefined matrix of clearing times is required to guarantee safe phase transitions. Furthermore, the consistency of the resulting signal plan needs to be checked before its application to ensure that, e. g., all signal groups are considered at least once within the cycle. Within this thesis, phase sequences are not optimised at run-time.

Offset The optimisation of offsets is necessary for the coordinated operation of neighbouring intersections, an aspect that is discussed in Chapters 7 and 8.

Equipping fixed-time controllers with an observer/controller to improve their performance and to reduce the effort for maintenance requires additional detectors (like inductive loops). If such detection hardware is available, traffic-actuated controls can be used.

5.2.2 Traffic-actuated controls

Since detectors are a prerequisite for the on-line adaptation of signal plans, the detector readings can also be used for local traffic-actuated control (see Section 2.3). Traffic-actuated controllers adapt their signalisation based on predefined temporal and logical conditions that represent the knowledge of the designing traffic engineer, but neither evaluate nor optimise their control strategy on-line.

By equipping a local traffic-actuated control with an additional observer/controller component, an on-line evaluation and optimisation can be realised. While the traffic-actuated controller adapts the intersection's signalisation, the temporal and logical conditions that define its operation are optimised on-line by the observer/controller. By using traffic-actuated controls within the SuOC, public transport prioritisation can be realised for organic intersections.

How a traffic-actuated controller's temporal and logical conditions are reconfigured on-line depends on the local TLC: When parameterisable industry-standard controls (like VS-Plus) are used, on-line learning and

optimisation will focus on the most relevant parameters available. These will typically include maximum phase durations as well as aspects of the phase extension process (like an extension step size or acceptable gaps). Constraints for the on-line reconfiguration of these parameters need to be defined by a traffic-engineer to ensure a safe operation of the organic intersection.

When a TLC is not only parametrisable but also programmable, the TLC's logic itself might be subject to on-line optimisation using Genetic or Evolutionary Programming (see Section 3.1.1). However, the on-line evolution of the logic is probably not feasible in practice due to safety issues.

5.2.3 Summary

The organic approach to intersection control proposed in this thesis relies on fixed-time or traffic-actuated TLCs that are extended by an observer/controller. Independent of the TLC's type, its configuration needs to suit the current traffic demand in order to reduce the vehicular delay and the number of stops at the intersection. In the following, the observer/controller architecture used for on-line reconfiguration is discussed in detail.

5.3 Observer

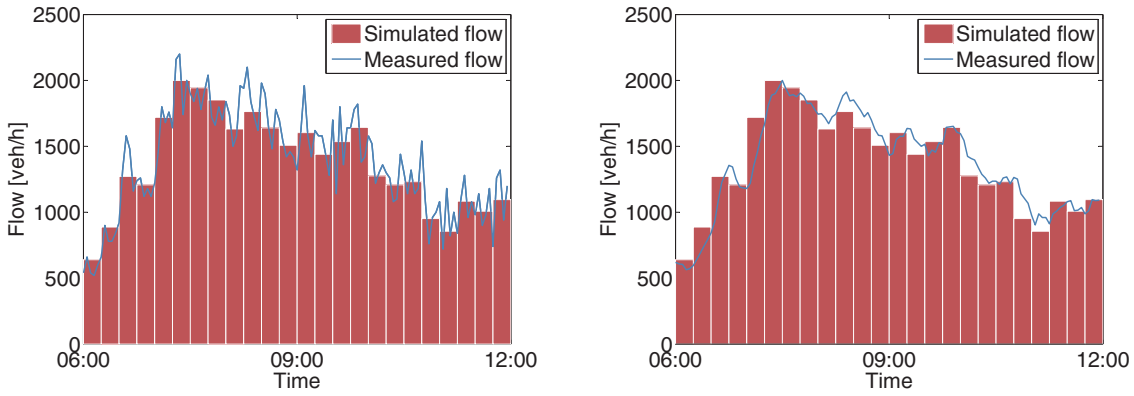
It is the observer's task to monitor and analyse the internal dynamics of the SuOC. At a signalised intersection, the observer monitors the current traffic demand and evaluates the active signal plan's performance. The resulting situation parameters are provided to the controller where they are utilised for the on-line selection and optimisation of signal plans. The observer comprises components for monitoring, preprocessing, data analysis, and prediction which are discussed in Sections 5.3.1 to 5.3.4. Section 5.3.5 focuses on the model of observation, before the observer functionality is summarised in Section 5.3.6.

5.3.1 Monitor and log file

The observer's monitoring component collects detection data from the controlled intersection and stores the resulting time series in a log file. For this thesis, it is assumed that flows and queues can be separately detected for the intersection's signal groups. Technical aspects of the required detection equipment (like the type or location of detectors) are not in the focus of this thesis, but will be briefly discussed in the context of preprocessing and data analysis.

5.3.2 Preprocessor

During preprocessing, the intersection's traffic flows are estimated from the monitored detection time series. The preprocessing results in a real-valued vector $M = (M_1, \ldots, M_n)$ containing the estimated hourly flows for each of the intersection's signal groups. Assuming that inductive loop detectors are available, these flows are extrapolated from the number of vehicles served within a rolling detection interval that considers the traffic light's last n_e cycles. The influence of the detection interval on the extrapolated flows is illustrated in Figure 5.2. For a signal group of a simulated intersection, the figure depicts the mean arrival rates configured in the simulation model as red bars. The blue lines show the extrapolated flows for $n_e = 2$ and $n_e = 10$, respectively.



(a) $n_e = 2$ (b) $n_e = 10$

Figure 5.2: Simulated and measured traffic flows for a signal group

When the detection interval is small, extrapolation puts a strong emphasis on recent vehicle arrivals (see Figure 5.2a). This allows for a quick recognition of changing demands, but tends to overemphasise random fluctuations around the mean arrival rate. Random fluctuations are handled better for a larger value of n_e, but this comes at the cost of a slower detection of changes in the demand (see Figure 5.2b). For the experiments conducted in this thesis, preprocessing is performed using $n_e = 10$. This choice keeps random influences acceptably small and yet allows for a sufficiently quick detection of changing demands.

5.3.3 Data analyser

The task of the data analyser is to derive performance measures for the intersection's signal plan, so the controller can update its internal mapping based on these real-world observations. Various measures can be evaluated, including vehicular delays, queue lengths, or stops (see Section 5.4). In contrast to the traffic flows derived by the preprocessor, most performance measures cannot be obtained directly from monitored detection time series, but need supplementing traffic models or approximation formulas.

Following the German *Handbuch für die Bemessung von Straßenverkehrsanlagen* (HBS, [69]), the average vehicular delay at the intersection is chosen as performance measure. An intersection's average delay t_D can be derived from the average delays of its signal groups using the equation

$$t_D = \frac{\sum_{i=1}^{n} (M_i \cdot t_{d_i})}{\sum_{i=1}^{n} M_i}, \tag{5.1}$$

where n corresponds to the number of signal groups, while M_i and t_{d_i} denote flow and delay for signal group i, respectively.

Unfortunately, the delay t_{d_i} of a signal group cannot be measured directly, but needs to be estimated by integrating the number of queued vehicles over time. Additional detection equipment (like upstream inductive loops) can be required to obtain the queue measurements. To overcome potential limitations in the detection process, a supplementary traffic model can support the delay measurements by estimating, e. g.,

queues beyond the detection range. However, such models are not in the focus of this thesis.

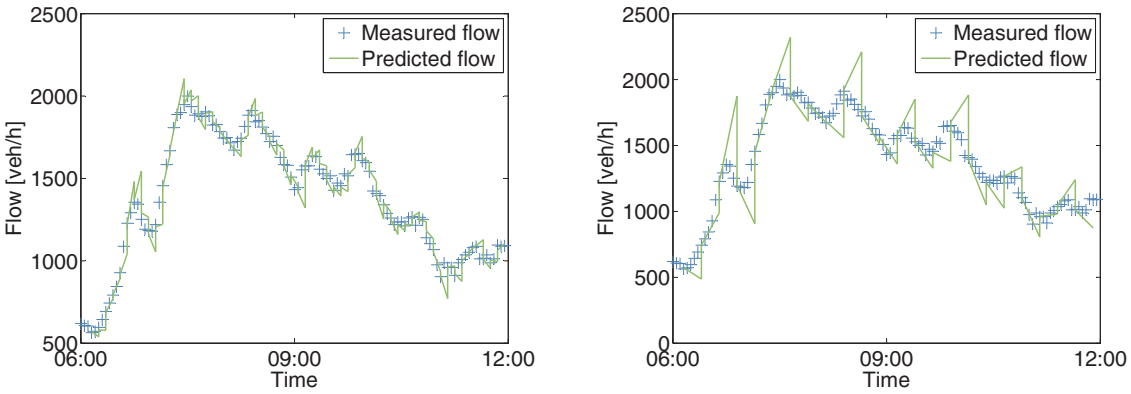
In case of fixed-time controlled intersections, Webster's formula (see Section 2.2.2) can approximate the vehicular delay without the need for additional detectors. For the approximation, only the current traffic flows and green times at the intersection are required. (The saturation flows used in Equation 2.2 are constant.) Flows are measured during preprocessing and green times are configured by the observer/controller, such that no additional detectors are necessary for delay approximation in case of fixed-time controls.

5.3.4 Predictor

The generic observer/controller architecture contains a prediction component to forecast future developments in the SuOC. At a signalised intersection, predictions refer to expected future traffic developments and can be based on current or historical traffic data [47]:

Current data A simple predictive approach that relies on current data only is to calculate a linear curve fitting of the latest traffic measurements. The extrapolated trend of this linear fit then serves as forecast. Linear fit predictions are well-suited for short-term predictions, but tend to loose their accuracy for longer prediction horizons (see Figure 5.3).

Historical data Predictions based on linear fits consider current traffic measurements only, but neglect knowledge on previous events. Using historical data, reoccurring traffic demands can be identified and grouped into classes. A class then represents reoccurring demands by a typical time series (see Figure 1.1). By comparing the latest traffic measurements to the available classes, the current demand can be automatically matched to the class that exhibits the smallest differences to the latest measurements. The time series representing this class is then used for forecasting. In contrast to linear fits, the use of historical data allows for long-term predictions. Therefore, a predictor might combine both approaches.



(a) Prediction horizon: 6 min (b) Prediction horizon: 15 min

Figure 5.3: Linear fit prediction

Besides the prediction techniques mentioned here, various other forecasting methods have been proposed [202]. Several of them can be utilised in the observer to obtain demand predictions that can supplement the preprocessor's flow measurements. However, to keep the observer simple, no prediction techniques have been implemented for this thesis.

5.3.5 Model of observation

In the generic observer/controller framework, a model of observation is included to configure the observer. The model specifies which SuOC properties are monitored and defines which analysis and prediction methods are applied. By limiting monitoring, data analysis, and prediction to a subset of the available data or methods, an efficient observation is possible even in case of limited resources (CPU and memory).

For an organic intersection controller, data analysis and prediction techniques can be selected in response to the controller's objective function (see Section 5.4.3). However, since sufficient computational resources for observation were available for the conducted experiments, an explicit model of observation has not been implemented for this thesis.

5.3.6 Summary

It is the observer's task to provide the controller with a set of situation parameters indicating the SuOC's state. For the proposed organic intersection controller, the situation parameters consist of two main components: The first component characterises the current traffic demand at the intersection by specifying the traffic flow for each signal group in a real-valued vector. The second component estimates the current vehicular delay at the intersection to judge the active signal plan's performance.

5.4 Controller

Based on the situation parameters received from the observer, it is the controller's task to provide the SuOC with an appropriate signal plan. The on-line signal plan selection is discussed in Section 5.4.1, while Section 5.4.2 is dedicated to off-line optimisations. Both processes require an objective function which is in the focus of Section 5.4.3, while Section 5.4.4 summarises the discussion of the controller functionality.

5.4.1 Level 1: Signal plan selection

This section introduces XCS-T, a novel classifier system that is designed for the on-line signal plan selection in the controller. XCS-T is based on the rule-based evolutionary on-line learning system XCS (see Section 3.2.1), but incorporates modifications that are necessary to meet the requirements of a learning signal controller.

The modifications include a real-valued knowledge representation and affect the discovery and exploration of classifiers: XCS creates classifiers by random covering and genetic modifications and evaluates their quality by performing exploration steps in the system environment. Random covering and environment-based exploration are only feasible in applications with a limited number of actions where the negative effects of exploration are limited. Since these preconditions are not fulfilled for traffic signal control, XCS-T replaces the covering mechanism with a simulation-based classifier creation (that is performed by the rule adaptation module) and introduces further necessary changes that are discussed in the following.

Knowledge representation

As part of an organic intersection controller, it is the task of XCS-T to provide a mapping that assigns appropriate signal plans to observed traffic demands. The mapping is learnt on-line based on the situation parameters provided by the observer and incorporates signal plans that have been optimised by Level 2 of the controller. Each classifier contained in the mapping consists of a condition matching a set of traffic demands, an action representing a signal plan, and some quality indicators estimating the classifier's goodness:

Condition A classifier condition specifies to which traffic demands a classifier is applicable. Assuming an intersection with n signal groups, the demand provided by the observer is a real-valued vector $M = (M_1, \ldots, M_n)$ containing the traffic flow of each signal group. To allow for continuous-valued inputs, XCS-T uses an *Ordered Bound Representation* [216]: A condition is a concatenation of interval predicates $[l_i, u_i]$, $i = 1, \ldots, n$, where $l_i, u_i \in \mathbb{R}$ are the lower and upper bounds of the intervals, respectively. A classifier matches an input $M = (M_1, \ldots, M_n)$ if and only if $l_i \leq M_i \leq u_i$ for all $i \in \{1, \ldots, n\}$. Therefore, the concatenation of intervals specifies an n-dimensional hyperrectangle containing all demands matched by a classifier.

Action A classifier's action corresponds to a signal plan. Signal plans can be represented differently, depending on the type of TLC used in the SuOC and the parameters under optimisation. When – like in Chapter 6 – the signal timings of a fixed-time controller are considered, an action corresponds to a vector specifying the phase durations.

Prediction, prediction error, and fitness The quality of a classifier can be judged from its prediction, prediction error, and fitness values. These values are updated based on the performance measurements provided by the observer, i. e., based on the vehicular delay t_D at the intersection.

Besides these core parts, each classifier comes with some additional bookkeeping parameters. These parameters are not discussed here in detail, since they serve the same purpose as in Wilson's XCS [213, 214].

Performance component

Using its population, XCS-T selects a signal plan suitable for the observed traffic demand following the process known from XCS (see Figure 3.3): Classifiers matching the observed traffic demand are selected from the population for inclusion in the match set. A prediction array is calculated and the signal plan with the best fitness-weighted delay prediction on average is chosen deterministically (exploitation). Classifiers advocating the chosen signal plan are transferred from the match set to the action set before the signal plan is returned to the SuOC.

Once activated at the intersection, the selected plan is kept active for an *activation interval* of at least n_c cycles before its performance (in terms of the resulting vehicular delay) is evaluated by the observer and reported to Level 1 of the controller. The activation interval needs to be selected carefully (see Section 6.5): It should not be overly long to allow for a fast reaction to changing demands, but it should be long enough to base the signal plan's evaluation on a reasonably long observation period. Based on the observed vehicular delay, the classifiers in the action set are updated by the reinforcement component of XCS-T.

Reinforcement component

The reinforcement component updates the prediction, prediction error, and fitness for all classifiers in the action set. Since the vehicular delay t_D for a signal plan is available as immediate reward, the XCS update mechanism for single-step problems is utilised. However, the delay should be minimised by XCS-T, while XCS is designed to select accurate classifiers with a maximal prediction. To obtain a maximisation problem, a payoff $P = c - t_D$ for a sufficiently large constant c is used (instead of $P = t_D$) when updating the action set. Prediction, prediction error, and fitness are then updated using Equations 3.1 to 3.3.

Discovery component

While the performance and reinforcement components of XCS-T are widely unchanged from XCS, substantial modifications are required for

the discovery component that is responsible for the creation and deletion of classifiers.

Creation of classifiers Although XCS is designed as on-line learning system, its classifier discovery process is unsuitable for an adaptive traffic control system due to several reasons:

Environment-based exploration In XCS, the quality of newly discovered classifiers created by random covering or genetic operators is explored in the system environment. However, in an adaptive traffic control system, a random creation of classifiers and the activation of their signal plans at the controlled intersection would result in drastically increased delays in the vast majority of cases. Creating classifiers randomly and performing explore steps in the SuOC is therefore no feasible option.

Simulation-based exploration Instead of relying on environment-based exploration, it is possible to perform exploration steps in simulation and exploit the learnt knowledge in the SuOC. Simulation-based exploration helps to judge the quality of classifiers without affecting the controlled intersection, but does not allow for a targeted classifier creation. When the number of available actions is large (like in signal control where thousands of signal plans can be selected), the creation of well-performing classifiers remains a time-consuming process when the standard discovery mechanism of XCS is applied.

Learning at design time Sometimes learning is performed in a test period at design time. Once XCS has "finished" learning, the learnt knowledge is merely exploited at run-time. Learning at design time is feasible only if classifiers covering all environmental inputs can be learnt during the test period. In a traffic control application, this assumption is not fully reasonable: While the traffic demands of regular weekdays can be learnt from simulations based on historical data, serious effort is required to learn signal plans covering long-term developments in traffic or unusual demands resulting from unforeseen events.

Since learning at design time does not suit the observer/controller architecture and exploration-based learning is not an option, the discovery component had to be modified. XCS-T does not rely on random covering or genetic operators for classifier discovery. Classifiers are created on demand instead: Whenever an observed traffic situation is not matched by any classifier in the controller's mapping (Level 1), the rule adaptation module on Level 2 (see Section 5.4.2) is activated to obtain an optimised signal plan including an estimation of its performance. The signal plan is then used to create a new classifier.

Let the unmatched traffic demand be denoted by $M = (M_1, \ldots, M_n)$. The new classifier's condition is designed to match demands similar to M. For $i = 1, \ldots, n$, the interval predicates $[l_i, u_i]$ are defined as

$$[\min(M_i - w_i, 0), \min(M_i + w_i, C_i)],$$

where $w_i \in \mathbb{R}^+$ is a *similarity tolerance* and C_i denotes the capacity of signal group i. The minimum calculated for the upper interval boundary u_i ensures that the classifier's signal plan is never operated in oversaturated conditions, while the similarity tolerance influences the classifier's applicability to varying traffic demands. The signal group-specific tolerance w_i is the product of the signal group's number of lanes and a lane tolerance w (given in veh/h). The lane tolerance w needs to be selected carefully (see experiments in Section 6.5): Small values create specific classifiers that match only few demands, resulting in a large number of classifiers and optimisations. Large values, on the other hand, will create overly general classifiers.

The new classifier's action comprises the signal plan optimised on Level 2. The initial prediction ρ_I is based on the plan's evaluation that has been provided by the simulation module. Thereby, the effect of the classifier is known although it has not yet been applied. The initial prediction error ε_I and fitness F_I are initialised pessimistically. Prediction, prediction error, and fitness are adapted by the reinforcement component after the classifier becomes part of an action set. Therefore, initial imprecisions in the quality indicators are corrected on-line.

Unfortunately, the evolution of signal plans in the rule adaptation component takes some time. Therefore, a new classifier for an unmatched demand $M = (M_1, \ldots, M_n)$ is not immediately available. To allow for an

immediate reaction of XCS-T in case of an empty match set, an existing classifier is selected from the population and its condition is widened:

Selection The signal plan of each classifier $cl \in [P]$ has been optimised for a specific traffic demand M_{cl} when it was created by the adaption module. As a signal plan that has been optimised for a demand similar to M is most likely to perform well in the current situation, the classifier system computes the Euclidean distance $d(M_{cl}, M)$ for each classifier $cl \in [P]$ and selects the classifier cl^* with the minimal distance for widening.

Widening To cover the unmatched demand M, the selected classifier cl^* is copied and widened. Assuming that cl^*'s condition is given by the interval predicates $[l_i^*, u_i^*]$, $i = 1, \ldots, n$, the predicates of the widened copy are replaced with

$$[\min(l_i^*, M_i), \max(u_i^*, M_i)].$$

Before the copy is included in the population, a capacity check ensures that its signal plan can handle the observed traffic demand. The check computes the capacity C_i of each signal group i considering the plan's (maximal) green times (see Section 2.2.2) and discards the widened copy if the traffic flow M_i exceeds C_i for any signal group. In this case, other classifiers $cl \in [P]$ with increasing distance $d(M_{cl}, M)$ are considered for widening. If no classifier in the population can be widened without violating the capacity constraint, a default signal plan is activated.

The widening of closely located classifiers enables an immediate response to an unknown traffic demand, while the capacity check ensures that the selected plans can handle the observed demand. However, widened classifiers potentially cause a suboptimal vehicular delay and their delay prediction becomes less accurate the more their condition is relaxed. The loss of accuracy is detected by the reinforcement component, but only after the classifier has been applied in the SuOC. The removal of (widened) classifiers with a relatively low accuracy is the task of the discovery component's deletion mechanism.

Deletion of classifiers The discovery component of XCS-T is not only responsible for the creation of classifiers, but also for their deletion. Classifiers are removed from the population on two occasions:

Subsumption deletion A classifier is deleted when it can be subsumed by a more general classifier that advocates the same action and is sufficiently accurate and experienced. Subsumption reduces the number of classifiers, but makes the classifier system more vulnerable to environmental changes as the more-specific, subsumed classifiers are not available as "fallback solution" [40]. As the population size reduction obtainable in the traffic control scenario is limited due to the large number of different signal plans, XCS-T does not apply subsumption deletion.

Population size limit When the population exceeds its size limit, XCS-T deletes classifiers in well-covered environmental niches that exhibit a lower than average fitness. The deletion mechanism is known from XCS (see Section 3.2.1).

With the discovery component that is responsible for the creation and deletion of classifiers, the on-line learning on Level 1 of the controller has been discussed completely. The optimisation of signal plans on Level 2 is in the focus of the following section.

5.4.2 Level 2: Signal plan optimisation

Level 2 of the controller supports the on-line signal plan selection by providing optimised signal plans. Optimisations are performed by the adaptation module that relies on the simulation module to evaluate the quality of candidate signal plans (see Figure 5.1).

Adaptation module

The adaptation module is implemented by an EA (see Section 3.1) or, more precisely, by the Evolution Strategy given in Algorithm 5.1. The genetic representation of signal timings, the variation and selection operators, and further aspects of the algorithm are discussed in the

following. The problem-specific configuration of the search heuristic is presented in Section 6.4.

Algorithm 5.1: Evolution Strategy (based on [9])

Generate μ initial parents and determine their objective function values.
repeat
  for λ *times* **do**
    Choose two parents at random.
    Recombine the selected parents.
    Mutate the preliminary offspring obtained from recombination.
    Determine the offspring's objective function value.
    Put the offspring into the offspring population.
  end
  Select the μ best individuals either from the offspring population ((μ, λ) strategy) or from the union of the parent and offspring population ($(\mu + \lambda)$ strategy).
  The selected individuals represent the new parents.
until *Stopping criterion is satisfied.*

Representation For optimisation, the signal timings for the controlled intersection need to be represented by a genotype. Following the work of Braun [29], a relative coding is used. Cycle time and phase durations of the signal plan are separately stored in a genotype vector $(x_0, x_1, \ldots, x_n) \in [0, 1]^{n+1}$, where x_0 codes the cycle length while x_i, $i = 1, \ldots, n$, represents the duration of phase i:

Cycle time The cycle time t_C is obtained from the genotype by interpreting x_0 as fraction of the difference between a maximum and a minimum cycle time, i. e.,

$$t_C = t_{C,min} + x_0 \cdot (t_{C,max} - t_{C,min}).$$

The maximum cycle time is assumed to be $t_{C,max} = 120\,\mathrm{s}$. The minimum cycle time $t_{C,min}$ depends on the intersection. It is

obtained by summing the minimum phase durations and the time t_T required for phase transitions.

Phase duration The phase durations are coded as fractions of the cycle. For each phase i, its duration d_i is calculated from the genotype as

$$d_i = d_{i,min} + \frac{x_i}{\sum_{j=1}^{n} x_j} \cdot \left(t_C - t_T - \sum_{j=1}^{n} d_{j,min} \right),$$

where $d_{i,min}$ is the minimum duration of phase i. The relative coding of phase durations ensures that each phase i is assigned its minimum green time $d_{i,min}$ and, additionally, a fraction of the green time available for distribution. The fraction corresponds to the phase's relative weight in the genotype, while the distributable green time is the portion of the cycle not reserved for phase transitions or minimum phase durations.

The relative coding has the advantage that each genotype represents a valid signal plan: The cycle length is guaranteed to lie in the interval $[t_{C,min}, t_{C,max}]$ while each phase is assigned at least its minimum duration. Therefore, no repair operators or penalty functions are required to handle invalid solutions. Furthermore, a predefined cycle length can be enforced easily which is especially useful for the coordinated operation of neighbouring intersections that is discussed in Chapters 7 and 8.

Evaluation function After decoding an evolved genotype to a signal plan, the plan's performance needs to be evaluated. Like on Level 1, the objective considered for evaluation is the vehicular delay at the intersection (see Equation 5.1). Delay estimates are provided by the simulation module which is discussed later in this section.

Selection mechanism Selection mechanisms are required for choosing parent individuals that undergo variation and for selecting survivor individuals that are carried over to the next generation:

Parent selection As typical for Evolution Strategies, parent individuals are randomly drawn from the population using a uniform distri-

bution [62, 209]. Thereby, parent selection is not biased by fitness values.

Survivor selection Survivor individuals are deterministically chosen based on their rank: After creating λ offspring and estimating their fitness, the best μ individuals are selected to form the next generation. Selection can consider only the offspring ((μ, λ) strategy) or might select the best individuals from the union of parents and offspring ($(\mu + \lambda)$ strategy).

Variation operators To create offspring from the parent individuals, the following variation operators have been implemented:

Mutation The mutation of a parent individual $x = (x_0, \ldots, x_n)$ is realised by adding a random noise Δx_i to each x_i, $i = 0, \ldots, n$. The value of Δx_i is randomly chosen according to a Gaussian distribution $N(0, \sigma)$ with zero mean and a standard deviation (or *step size*) of σ. Mutating the parent x results in an offspring $x' = (x'_0, \ldots, x'_n)$ with $x'_i = x_i + N(0, \sigma)$.

Using a *self-adaptive Evolution Strategy*, the standard deviation σ is included in the evolutionary process. As *strategy parameter*, it becomes part of each individual and is mutated in each time step. Therefore, the mutation mechanism is specified by adapting σ using the formula

$$\sigma' = \sigma \cdot e^{N(0,\tau)}$$

and mutating the individual afterwards using the obtained σ' as standard deviation, i. e.,

$$x'_i = x_i + N(0, \sigma').$$

Following [62], the *learning rate* τ for the strategy adaptation is chosen inversely proportional to the square root of the problem size, i. e., $\tau = 1/\sqrt{n+1}$.

Crossover When recombining two parent individuals $x = (x_0, \ldots, x_n, \sigma_x)$ and $y = (y_0, \ldots, y_n, \sigma_y)$ to create one offspring $z = (z_0, \ldots, z_n, \sigma_z)$, *discrete recombination* is used for the signal timings, i. e., for each

position $i \in \{1, \ldots, n\}$, z_i is randomly chosen as x_i or y_i. The strategy parameters are recombined using *intermediary recombination*, i. e., $\sigma_z = (\sigma_x + \sigma_y)/2$. Discrete recombination preserves the diversity within the phenotype space, while intermediate recombination ensures a cautious adaptation of strategy parameters [62].

Further aspects Further aspects include the creation of a start population, the determination of suitable sizes for the parent and offspring population, and the selection of a stopping criterion. All choices aim at finding (near-)optimal signal timings in a minimum amount of time:

Start population The start population is initialised randomly, but might include the best known signal plan (provided by XCS-T) or – if applicable – a plan calculated according to Webster's method (see Section 2.2.2) to guide the evolutionary search.

Population size The ratio of the parent and offspring population size determines the selection pressure. For (μ, λ) selection, the μ/λ ratio is typically in the range of 1/5 to 1/7 [62, 209]. For $(\mu + \lambda)$ selection, it can be sufficient to create a single offspring (resulting in a steady-state Evolution Strategy), but usually λ is at least as large as the parent population (i. e., $\lambda \geq \mu$). The minimum size of the parent population depends on the optimisation problem. Together with the selection pressure, it is investigated in a sensitivity study (see Section 6.4).

Stopping criterion Since the optimised signal timings should be made available to the selection mechanism quickly, a maximum number of evaluations is chosen as stopping criterion.

Simulation module

An aspect that has not yet been discussed is the fitness evaluation of evolved signal plans by the simulation module. The module is configured according to the traffic demand measured in the SuOC and considers the signal timings that are specified by the candidate plan. In this thesis, evaluations are performed using a microscopic traffic simulator or, alternatively, Webster's approximation:

Microscopic simulation A microscopic simulator (see Section 2.5.1) is the most general way to evaluate the quality of a candidate signal plan. This thesis uses AIMSUN (*Advanced Interactive Microscopic Simulator for Urban and Non-Urban Networks*, [11, 45]) as simulation model. In AIMSUN, fixed-time and traffic-actuated controls can be simulated and a variety of objectives is available for evaluation (including measures that are difficult to obtain outside of a simulated environment). However, a price to pay for this generality is that simulation-based evaluations are noisy and relatively time-consuming. In consequence, several questions regarding the handling of stochastic results need to be investigated, including the number of repeated simulations and their duration. These issues will be discussed in Section 6.3.

Webster approximation When available, approximation formulas are an alternative to microscopic simulations since they can be computed significantly faster than simulations and provide deterministic results. Unfortunately, approximation formulas are available for some special cases, only. In Chapter 6, the average delay at a fixed-time controlled intersection is estimated by Webster's formula that has been introduced in Section 2.2.2.

With AIMSUN simulations and Webster approximations, two alternative evaluation mechanisms are available for the experiments conducted in this thesis. Before the experimental evaluation is discussed in Chapter 6, some general thoughts on possible objective functions are presented in the following.

5.4.3 Objective function

Both levels of the controller require an objective function to guide the selection (Level 1) and optimisation (Level 2) of signal plans. The objective function needs to be defined by a traffic engineer. In its most general form, it is given as *performance index*

$$PI = \sum_{i=1}^{n} \omega_{i,1} o_{i,1} + \omega_{i,2} o_{i,2} + \cdots + \omega_{i,m} o_{i,m},$$

where n corresponds to the number of signal groups and m to the number of objectives. The variables $\omega_{i,j}$ and $o_{i,j}$ specify the weight and value of objective j for signal group i, respectively.

Following [69], the average vehicular delay at the intersection is used as performance index within this thesis. It is obtained by summing the average delays for the intersection's signal groups weighted by their relative flows (see Equation 5.1). However, the performance index is not limited to delays. Other objectives can be included, the only prerequisite is that they can be measured by the observer or estimated by its data analysis component. By adapting an objective's weight, its importance can be adjusted for the different signal groups. Whenever a signal group has, e. g., a limited queuing space, its queues can be heavily weighted in the performance index. Furthermore, the performance index can model derived objectives as weighted combination of available measures, e. g., the vehicular fuel consumption can be approximated from available delays and stops.

Although several objectives can be included in its definition, the performance index itself constitutes a single-objective function. All considered objectives are combined to a single fitness value using a weighted sum. The reason for this restriction is in the automatic selection of signal plans performed on Level 1 of the controller: Multi-objective functions that treat several contradicting objectives separately have a set of Pareto-optimal solutions which represent different trade-offs among the objectives. To automatically select a single signal plan from this Pareto-optimal set, the importance of the objectives needs to be weighted. The weights can either be defined explicitly (like with a single-objective function) or implicitly by selecting the Bellman-Zadeh solution [18] which is located in the centre of the Pareto-optimal set. Since the necessity of weights in the automatic signal plan selection limits the benefits of multi-objective functions, the performance index is restricted to a single fitness value.

5.4.4 Summary

Based on the situation parameters provided by the observer, the controller evaluates and reconfigures the intersection's signalisation. Signal plans are optimised based on simulations, their quality for different demands

is learnt on-line. To perform learning and optimisation, the controller has a two-levelled structure: On Level 1, XCS-T (a classifier system that is based on Wilson's XCS) maps signal plans to traffic demands. The mapping is continuously updated using performance evaluations provided by the observer. On Level 2, signal plans are optimised by an EA based on traffic simulations. The simulation-based optimisation allows to safely explore candidate plans.

The combination of on-line learning and off-line optimisation is novel. It allows for autonomous learning in scenarios where complex actions (like signal plans) have to be learnt while exploration in the environment is infeasible. Two-levelled learning is therefore not limited to signal control, but is applicable to a wide range of applications. The combination of a classifier system for on-line learning and an EA for off-line optimisation proposed here is the first successful implementation of two-levelled learning for a technical system.

5.5 Implementing organic intersections

The previous sections introduced an adaptive intersection controller with on-line learning capabilities that is based on the observer/controller framework. In the following, the controller's deployment at a real-world intersection is discussed (Section 5.5.1) and its implementation for the conducted simulation studies is presented (Section 5.5.2).

5.5.1 Real-world deployment

The proposed observer/controller architecture for signalised intersections advances the state of the art because of its on-line learning and optimisation capabilities. Although this thesis focuses on these organic properties and not on issues related to the architecture's real-world deployment, the principal applicability of the proposed observer/controller at a real-world intersection is briefly discussed here. Issues that need to be considered include the hardware requirements for detection and computation as well as the operational safety of on-line learning.

Detection requirements

It is the observer's task to monitor the intersection and to provide its traffic demand and signal plan performance to the controller: The measured demand is characterised by the average flow of the intersection's signal groups for some time period in the past. Assuming a separate detection for each signal group, the required flows can be determined using inductive loop detectors.

The performance evaluation of a signal plan is more complicated since it is based on the vehicular delay at the intersection. A delay estimate cannot be obtained directly from the intersection's detectors, but requires a traffic model that sums up the waiting times for the queued vehicles. Therefore, queues at the intersection need to be detectable which requires a combination of stop-line and upstream detectors (when using inductive loops) or more sophisticated detection devices (like video detectors). In case of fixed-time controls, Webster's formula (see Section 2.2.2) can be used to obtain a delay estimate from the intersection's traffic flow and signal timings. For the necessary flow measurements, a single inductive loop per signal group is sufficient.

Computational requirements

Considering the hardware required to operate the observer/controller, an embedded PC that is based on energy efficient hardware is assumed to be available at the controlled intersection.

The SuOC observation – that involves the calculation of average flows, the estimation of delays, and possibly traffic predictions – is not computationally demanding. This is also true for the signal plan selection performed on Level 1 of the controller which can be computed quickly for reasonable mapping sizes. However, the optimisations on Level 2 can be demanding when evaluations are based on microscopic simulations. Here, the computational power of an embedded PC is required to guarantee a reasonable optimisation time of a few minutes.

Regarding the embedded PC's energy consumption, it should be noted that the vehicles' reduced fuel consumption and pollution emissions will justify the additional effort.

Safety of operation

Not only technical feasibility, but also operational safety is important when discussing the deployment of an organic intersection controller. It has to be guaranteed that the adaptation and learning capabilities of the controller cannot result in an invalid signal plan: Conflicting traffic movements must not be assigned the right of way simultaneously, all signal groups have to be included in the cycle with at least their minimum green time, clearing times between signal groups have to be kept, and so on. To this end, it is important to notice that the built-in safety mechanisms of a TLC remain functional when it is reconfigured by the observer/controller. The observer/controller is merely an extension that provides learning and optimisation capabilities, but is not required for the operation of the signalised intersection.

Regarding learning and optimisation in the observer/controller, a traffic engineer needs to specify parameters that are considered for on-line adaptation. For the selected settings, feasibility constraints (like allowable phase transitions or clearing times) have to be provided. Signal plans learnt by the observer/controller can be guaranteed to be valid with respect to the given constraints. This allows for operational safety.

Summary

Regarding its detection and hardware requirements, the proposed observer/controller architecture is applicable in a real-world setting: While its detection requirements are similar to those of a traffic-actuated controller (see Section 2.3), the observer/controller requires more powerful hardware to perform signal plan optimisations. Such hardware is available, the effort required for its integration and its energy demand are expected to be compensated by the reduced fuel consumption and pollution emission at the controlled intersection.

To guarantee the operational safety of an organic intersection, a traffic engineer has to select the TLC parameters considered for on-line adaptation and needs to specify constraints on their feasibility. The on-line reconfiguration of signal plans is then guaranteed to comply to the given constraints.

5.5.2 Simulation-based evaluation

The experiments conducted in Chapters 6 to 8 have not been performed in a real-world traffic network, but are based on simulations. This section briefly outlines technical aspects of the underlying implementation and discusses issues resulting from the use of a simulated SuOC.

Technical aspects

In the conducted simulation studies, the microscopic traffic simulator AIMSUN serves as SuOC and as simulation module within the observer/controller.

AIMSUN as part of the SuOC As part of the SuOC, AIMSUN provides a simulated traffic network that substitutes its real-world counterpart. A Java [91] implementation of the observer/controller architecture monitors and controls the simulated SuOC on-line. To this end, AIMSUN provides a programming interface – the *AIMSUN API* – that allows to read data from a running simulation and to influence the elements of the simulation model. Using the API, it is possible to obtain data like a detector time series from a running simulation or to adapt the timings of a traffic signal. AIMSUN's API is available for the C/C++ [184] and Python [16] programming languages, but not for Java. Therefore, the simulation model cannot be directly accessed from the observer/controller.

To monitor and control the simulated SuOC, an API module has been implemented in C/C++ that obtains the current state of AIMSUN's network elements in every simulation step (see Figure 5.4). Using the Java Native Interface [120], a Java representation of the simulated network is created. By accessing this representation, the observer/controller can monitor and control all elements of the simulated network. The simulation progress is instantly reflected in the network's Java representation, while changes made by the controller are reflected in the simulated network in the next simulation step.

AIMSUN as part of the observer/controller Within the observer/controller architecture, the rule adaptation module on Level 2 of the controller

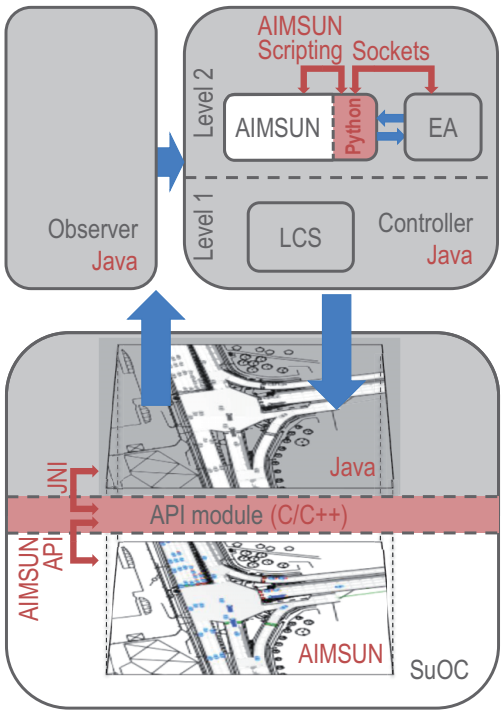


Figure 5.4: Interface between AIMSUN and observer/controller

can be supported by a simulation component. In the experiments conducted for this thesis, the core of this module is an EA that requires quality estimates to evaluate the evolved signal plans. These estimates can be obtained from simulations.

When the evaluation of candidate solutions is based on traffic simulations, the EA makes use of AIMSUN as the controller's simulation component (see Figure 5.4). For automated simulations, AIMSUN provides a *Scripting Interface* for Python that allows to open a network model, to modify the model's parameters (like traffic demands or signal timings), and to start a simulation.

Unfortunately, AIMSUN's Scripting Interface cannot be accessed directly from the EA, since the observer/controller has been implemented in Java. Therefore, the EA communicates traffic demands and candidate signal plans to a Python script using a communication socket. The script configures the simulated network accordingly using AIMSUN's Scripting Interface, starts the simulation run, and communicates the results to the EA.

Issues resulting from a simulated SuOC

Using a simulated SuOC to evaluate the proposed observer/controller architecture for traffic control requires careful considerations to ensure that obtained results can be carried over to real-world networks.

Availability of data Microscopic traffic simulators offer a wide variety of statistical information on the simulated network and its elements, but only a subset of this data is available from detectors installed in the real-world network. To ensure that the proposed organic approach remains applicable in the real-world, it is important to restrict the observer/controller's monitoring to data obtainable from real-world detectors.

Considering the proposed observer/controller architecture, the required traffic measures can be obtained in a real-world setting (see Section 5.5.1). In the conducted simulations (see Chapter 6), the observer therefore monitors only data detectable at a real-world intersection. However, it is important to note that the mentioned restrictions apply only to the observer/controller, but not to an evaluation performed after the simulation run. Here, all statistical data provided by the simulator can be used including, e. g., data on the fuel consumption or pollution emission of vehicles that is not available to the observer/controller in its decision process.

Validity of simulations When a simulated SuOC replaces a real-world network, it is important that the simulation resembles the real-world with sufficient accuracy. Since microscopic simulators in general and AIMSUN in particular are successfully used by researchers and practitioners in a wide range of applications including the evaluation of signal systems [57, 99, 132], it seems reasonable to consider microscopic simulations as sufficiently exact for the experiments conducted here.

Although a calibration of model parameters is usually required to reproduce a network's traffic dynamics in great detail (see Section 2.5.1), the experiments presented in Chapters 6 to 8 are based on uncalibrated network models. As this thesis does not aim at creating an optimal intersection controller for a specific real-world intersection or network, but focuses on evaluating the adaptation and learning capabilities of the

observer/controller architecture, the use of uncalibrated network models seems justifiable here.

Time synchronisation Assuming a sufficiently accurate simulator, a simulated SuOC has important advantages: The observer/controller components and their configuration can be safely tested and evaluated. Furthermore, the time requirements for experimentation can be reduced, since microscopic simulations can be performed at different speeds including simulations that run significantly faster than wall-clock time. However, it is important to synchronise the simulation of the SuOC with the observer/controller. Therefore, the simulated SuOC runs in wall-clock time when optimisations are performed on Level 2 of the observer/controller architecture. This ensures that the same amount of time passes in the simulated SuOC as it would pass at a real-world intersection during optimisation.

Dual use of simulator Another issue arises from the dual use of AIMSUN in the SuOC and as simulation module within the observer/controller, since the simulation module might resemble the SuOC too closely. To avoid that results from the simulation module are more exact than they would be at a real-world intersection, countermeasures are taken: Firstly, the simulation module is configured based on traffic flows detected within the SuOC. The traffic demands defined in the model (which are accessible only in simulation) are not taken into account (see *Availability of data*). This restricted use of data resembles the real-world situation. Secondly, the SuOC and the simulation module make use of different random seeds to ensure stochastic differences between both simulations (see Section 2.5.1).

Summary

When using a simulated SuOC to evaluate the proposed observer/controller architecture, several issues need to be considered to ensure that obtained simulation results carry over to a real-world setting. These issues include the availability of detection data at a real-world intersection or the time synchronisation of SuOC and observer/controller. The

conducted simulation studies keep these issues in mind. With the necessary precautions, a simulated SuOC allows for a relatively fast testing of different implementation alternatives, while the findings obtained from the simulation-based evaluation of the observer/controller are expected to reflect the results from a real-world setting with sufficient accuracy.

CHAPTER 6

Experimental validation of organic intersections

A simulation study has been conducted to evaluate the performance of the organic intersection controller presented in the previous chapter. As test case, the study considers two intersections located at Hamburg, Germany. Both are presented in Section 6.1.

Section 6.2 is dedicated to general aspects regarding the design of experiments and provides an overview of the sensitivity study that has been conducted to determine a reasonable configuration of the observer/controller architecture. The study pays special attention to the architecture's adaptation and learning capabilities: Sections 6.3 and 6.4 discuss the configuration of the simulation module and the EA which in combination form the controller's optimisation level. The configuration of the LCS responsible for signal plan selection is in the focus of Section 6.5.

Using the results of the sensitivity study, the organic controller has been experimentally evaluated. Results of the simulation-based evaluation are discussed in Section 6.6.

6.1 Test case

In the evaluation of the observer/controller architecture, two intersections located at Hamburg, Germany, serve as test case:

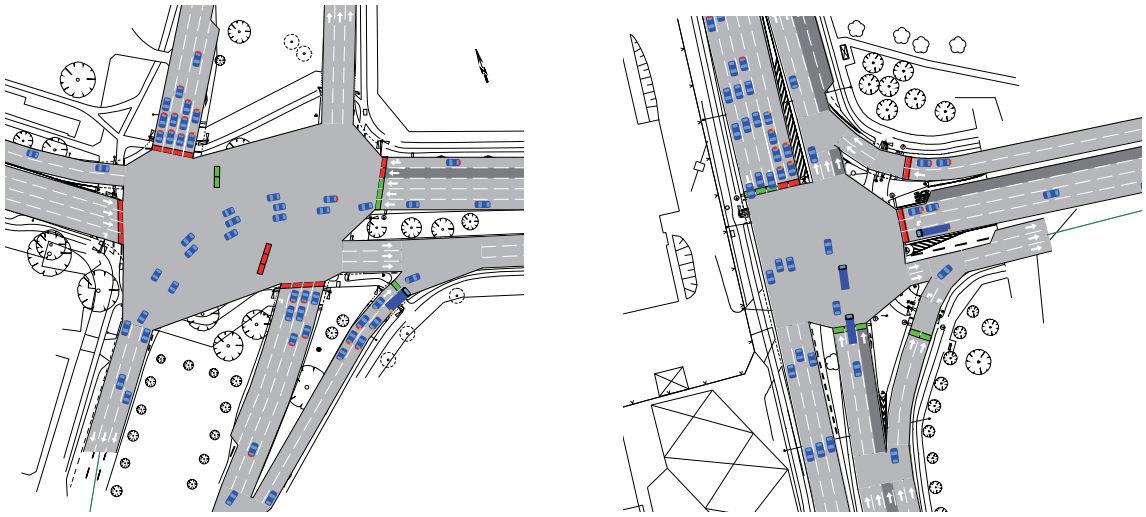
Alsterkrugchaussee / Deelböge / Borsteler Chaussee (K3) Intersection K3 is a four-armed junction that is equipped with 25 signal groups, ten of which serve motorised traffic movements. For the traffic-actuated control of a subset of these signal groups, six inductive loop detectors are available.

Kollaustraße / Nedderfeld (K7) Intersection K7 is a three-armed junction that is controlled by a fixed-time controller. The intersection is equipped with 17 signal groups, seven of which serve motorised traffic movements.

Simulation models of both intersections have been created in the microscopic traffic simulator AIMSUN [11, 45] based on detailed site plans of the real-world junctions (see Figure 6.1). In the models, traffic demands are programmed for a time period beginning at 6 a. m. and ending at 7 p. m. The simulated demands are taken from a traffic census that has been kindly provided by *Schmeck Ingenieurgesellschaft mbH*. For a regular workday, the census reports vehicle counts for the intersections' turnings with a resolution of 15 min. Furthermore, the vehicle types "car" and "truck" are distinguished. Figure 6.2 summarises the demands for K3 and K7.

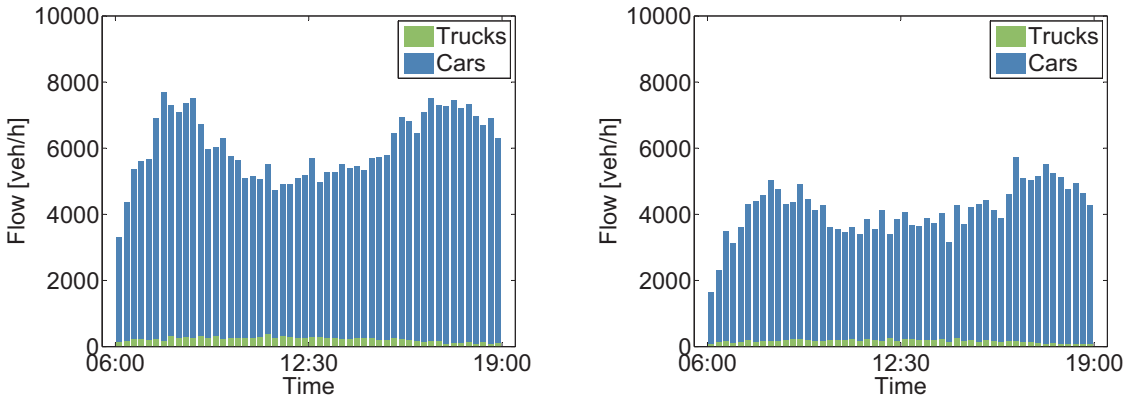
Reference signal plans for comparison have been provided by *Landesbetrieb Straßen, Brücken und Gewässer (LSBG)*, Hamburg. At both intersections, signal plans are changed based on the time of day. From 5 a. m. to noon, intersections are controlled by a morning plan that is replaced by an afternoon programme running till 8 p. m. (7 p. m. on Fridays). For both, morning and afternoon plans at Intersections K3 and K7, signal groups are served in several phases such that Webster's green time optimisation technique (see Section 2.2.2) is not applicable. Webster's delay formula (Equation 2.3) can be applied to estimate the effect of changed signal timings on the vehicular delay, though.

In the simulations, both intersections have been modelled as fixed-time controlled. For the traffic-actuated Intersection K3, the simulated fixed-



(a) K3 (b) K7

Figure 6.1: Simulation models of Intersections K3 and K7



(a) K3 (b) K7

Figure 6.2: Traffic demands for Intersections K3 and K7 (Data provided by Schmeck Ingenieurgesellschaft mbH)

time plan assigned the maximal extension to the actuated signal groups. Since a subset of at most four signal groups is affected, this simplification is assumed to have only a minor influence on the performance of the reference plan. The simplification is necessary, since the simulation of a traffic-actuated control based on a proprietary control logic would have required an additional plug-in for the AIMSUN simulation environment that was not available.

Before the organic intersection controller is evaluated using the simulation models of K3 and K7 as test case, a reasonable configuration of the observer/controller architecture has been determined in a sensitivity study. The study and all evaluations have been conducted using AIMSUN v. 5.1.11 running under Microsoft Windows Vista 64-bit. A single core of a 2.5 GHz Intel Core 2 Quad processor equipped with 8 GB RAM served as computing platform for the observer/controller. Its evaluation followed established experimental design guidelines that are briefly recapitulated in the following section.

6.2 Experimental design

When it comes to experimentation, it is important to have a clear idea of what is to be studied and how the experiment is to be designed and analysed. Therefore, Section 6.2.1 reviews general guidelines for the design of experiments before Section 6.2.2 provides an overview of the conducted sensitivity study.

6.2.1 Guidelines for experimental design

Regarding the design of experiments, Montgomery [133] proposes the following general guidelines for pre-experimental planning, experimentation, and data analysis:

1. *Problem statement* When planning an experiment, it is important to have a clear idea of what is to be studied. Preparing a list of problems or questions that should be addressed by the experiment contributes to a better understanding of the studied phenomenon or system. Furthermore, the experiment's overall objective should be

kept in mind: An experiment can aim at *factor screening* or *factor optimisation* to characterise and configure factors that influence the system performance. Other objectives include the *confirmation* of assumed system properties or the study of a system's *robustness* with respect to the variation of factor levels.

2. Response variables A system's output or performance is characterised by a set of response variables. In pre-experimental planning, response variables that provide useful information with respect to the problem statement need to be identified. Since variables or their measurements can be noisy, response variables can also include the average or standard deviation of a system characteristic.

3. Factors, levels, and ranges Factors that may influence the performance of a system or process can be classified as either design factors or nuisance factors. *Design factors* are factors selected for study in the experiment, while *nuisance factors* are not of interest (or are uncontrollable) in the current experiment although they may affect the response variables.

When the design factors are identified, the *ranges* over which the factors will be varied and the specific *levels* considered in the experiments have to be defined. The experiment's overall objective influences the choice of levels and ranges: For factor screening, ranges over which the design factors are varied are usually broad, while the number of factor levels is kept small. When the screening is finished, a subsequent factor optimisation might investigate the configuration of important design factors with smaller ranges and more levels.

4. Experimental design When the previous steps of pre-experimental planning have been performed, an experimental design needs to be chosen. The design specifies how the factors are varied in a sequence of runs. Although *one-factor-at-a-time* designs are common, they do not consider interdependencies among the factors. Therefore, *factorial experimental designs* that vary several factors at a time constitute a better approach. However, factorial designs for k factors, each at two levels, generally require 2^k runs, i. e.,

the number of runs grows rapidly. Therefore, *fractional factorial designs* that use only a subset of runs are another important type of experimental design.

5. *Experiment* The experiment is conducted following the chosen experimental design.

6. *Data analysis* To draw objective conclusions, data derived in the experiment can be analysed using *graphical methods* or *statistical techniques* like hypothesis testing or confidence interval estimation. Sometimes, an *empirical model* that expresses the relationship among important design factors and response variables can be obtained.

7. *Conclusion and recommendation* Based on the results of data analysis, conclusions regarding the problem statement and recommendations on further actions are given. Often, recommendations lead to subsequent experiments for confirmation testing or for the refinement of a hypothesis. Therefore, experimentation is an iterative process.

The guidelines for planning, conducting, and analysing experiments serve as blueprint for the sensitivity study that has been conducted for the observer/controller.

6.2.2 Sensitivity study

The observer/controller framework for signalised intersections allows for the autonomous reconfiguration of signal plans at run-time. The architecture thereby frees the traffic engineer from the often tedious task of specifying several signal plans for typical traffic demands. However, the observer/controller itself is a complex system that needs to be configured once before it can autonomously adapt the signalisation of various intersections.

Figure 6.3 depicts a simplified cause-effect diagram for the observer/controller architecture. Grouped according to the architecture's main components, the figure shows selected factors influencing the performance of an organic intersection. While the factors listed for the SuOC are

uncontrollable except for the signal plan, the factors listed for the observer/controller are potential design factors.

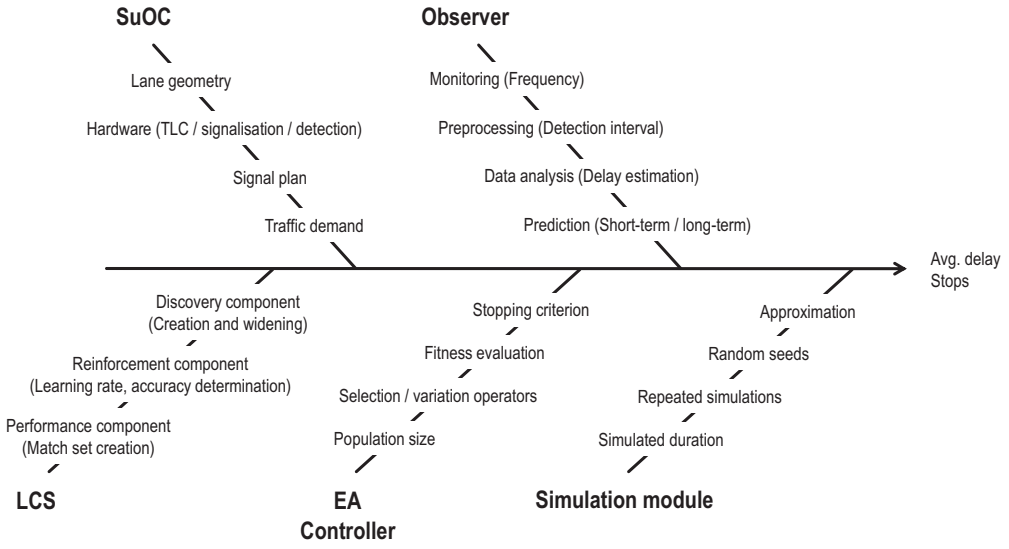


Figure 6.3: Simplified cause-effect diagram for the observer/controller

To determine a reasonable configuration for the most important design factors, a sensitivity study has been conducted. The study focuses on the controller and starts with the configuration of the simulation module.

6.3 Simulation module

The simulation module estimates the fitness of signal plans to guide the evolutionary search. It thereby influences the quality of evolved solutions and the time required for an optimisation run. In this thesis, fitness estimations are based on microscopic simulations that allow to evaluate fixed-time and traffic-actuated signal plans with respect to a wide variety of objectives. However, this flexibility comes at a cost:

- Microscopic simulations are noisy. The obtained quality estimates do not depend on the effectiveness of the signal plan alone, but are influenced by the stochastic nature of the simulation.
- Microscopic simulations have high run-time requirements, especially when accurate estimations are required.

The trade-off between estimation quality and the time required for simulation can be influenced by the simulator's configuration in a number of respects:

Simulated duration While extensive simulations can reduce the influence of stochastic fluctuations and can therefore provide a more reliable fitness estimation, they increase the necessary execution time for an evaluation. The trade-off between estimation quality and execution time caused by different simulated durations is investigated in Section 6.3.1.

Repeated simulations In a stochastic simulation, the random seed determines the outcome of all random choices during the simulation period and therefore influences the simulation results. A way to reduce deviations is to average the results of several simulation runs that use different random seeds. Section 6.3.2 investigates whether it is beneficial to average several short simulations instead of running a single but time-consuming simulation.

Simulation seed When comparing signal plans based on the outcome of stochastic simulations, differences in the observed performance are not only due to different signal timings, but can be partly attributed to fluctuations in the experimental conditions (e. g., when simulations exhibit different vehicle arrival patterns). A common random seed for the simulator can help to keep such fluctuations at a minimum. Therefore, the benefit obtainable by comparing signal plans based on common random numbers will be investigated in Section 6.3.3.

6.3.1 Simulated duration

In microscopic simulations, the estimation quality that can be obtained from a simulation run depends on the run's simulated duration that can be subdivided into an optional *warm-up period* and the mandatory *simulation period*. During warm-up – which precedes the simulation period – traffic is normally simulated, but no traffic statistics are recorded. The warm-up period fills the initially empty network with traffic such

that results gathered during the succeeding simulation period are not influenced by start-up effects.

To determine the minimal simulated duration that allows to accurately estimate a signal plan's quality, the relation of simulated duration, execution time, and result variability is in the focus of a first experiment. A second experiment investigates the simulated duration's influence on the correctness of signal plan rankings.

Execution time and standard deviation of simulation results

The relation of simulated duration, required execution time, and the resulting standard deviation of delay estimates is investigated in a factor screening experiment that considers the duration of the warm-up and simulation periods as design factors.

Simulations have been conducted using no warm-up period, a warm-up of 900 s, or a warm-up of 1800 s, while the simulation period has been varied between 1800 s and 14 400 s in steps of 1800 s. For each factor combination, 100 replications using different random seeds have been simulated. Furthermore, signal plans in under- and oversaturated traffic conditions have been considered, since microscopic traffic simulations are known to exhibit a larger variability when flow is close to, or exceeds, capacity [104].

As response variables, the average delay measurements, their standard deviation, and the required execution time have been investigated. The goal of the experiment is to determine factor combinations for signal plans in under- and oversaturated conditions that provide precise delay estimates, but require minimal execution time.

Signal plans in undersaturated conditions As example of a signal plan operating in undersaturated conditions, the morning plan of Intersection K3 has been simulated for the traffic demand observed at 11 a. m. Figure 6.4 shows the influence of different warm-up and simulation periods on the execution time required for 100 fitness evaluations. Execution time measurements have been used for a linear regression, the resulting regression functions are shown in the figure.

The observed linear relation of simulated duration and execution time has been expected, since the time requirements of a microscopic simulator

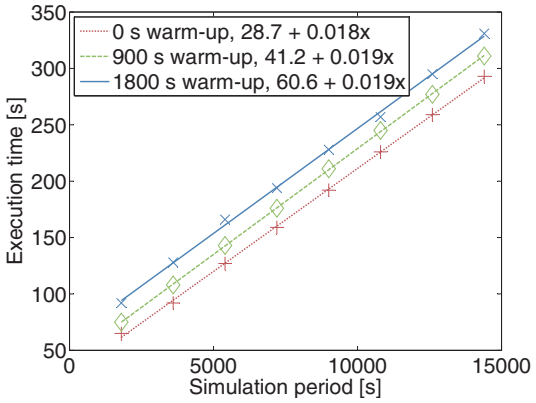
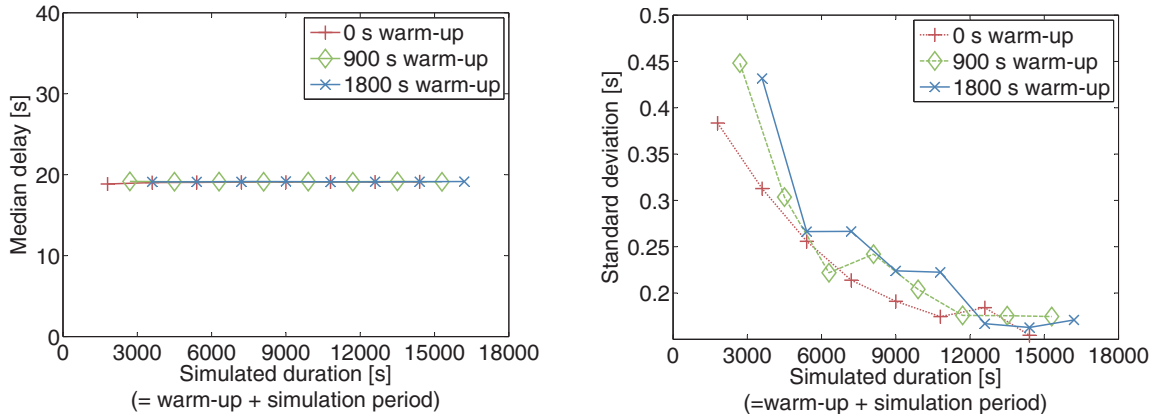


Figure 6.4: Influence of the simulation period on the execution time

strongly depend on the number of vehicles that are handled in each simulation step (see Section 2.5.1). However, it should be noted that there is an overhead for starting the simulation and simulating the warm-up period. The overhead corresponds to the y-intercept of the regression functions which increases approximately linearly with the warm-up duration.

Besides execution time requirements, the obtainable quality of results is an important aspect. To this end, Figure 6.5 depicts the median delay and the standard deviation of the delay measurements at Intersection K3 for each investigated factor combination. Qualitative statements derived from the figure are expected to carry over to other intersections in undersaturated conditions.

Figure 6.5a shows that the median delay remains widely unaffected by the simulated duration. Delay measurements are randomly distributed around the true average delay even for short warm-up and simulation periods. However, the simulated duration does affect the standard deviation of the simulation results (see Figure 6.5b). As expected, longer simulation periods lead to more consistent results and reduce the influence of the simulation's random seed. However, the standard deviation cannot be linearly reduced with the simulation period. When using microscopic simulations as fitness function, it is therefore important to select a sufficiently large simulation period to benefit from the initially strongly reduced deviations, but to avoid overly long simulations unless reduced random noise is much more important than execution time.

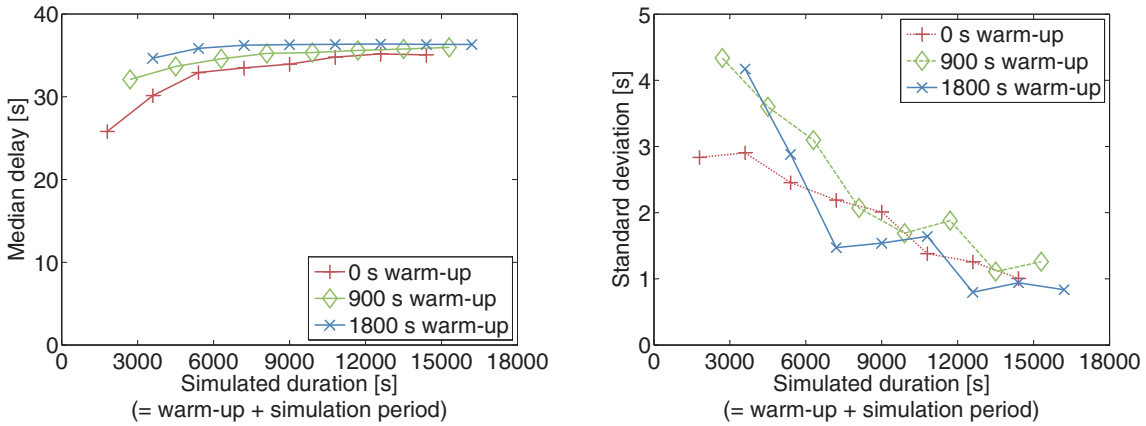


(a) Median of measured delays (b) Standard deviation of measured delays

Figure 6.5: Influence of the simulated duration on measured delays for a signal plan in undersaturated conditions (based on 100 replications)

Regarding the use of an additional warm-up period, Figure 6.5b does not indicate a beneficial effect on the standard deviation of measured delays. This is probably due to the small size of the test network that consists of a single intersection, only. In undersaturated conditions, such small networks quickly fill with traffic, making extensive warm-up periods obsolete. With respect to the execution time, an additional warm-up period is nearly as costly as a regular simulation period of the same length, since the same simulation model is used during warm-up, but no traffic statistics are recorded. Therefore, warm-up periods are not beneficial in undersaturated conditions and should be avoided when the simulated network consists of a single intersection.

Signal plans in oversaturated conditions The previous experiment has been repeated for oversaturated conditions by simulating the morning plan of Intersection K3 during the morning peak hour. Again, the length of the warm-up phase and the simulation period have been considered design factors, while the average delay measurements and their standard deviation have been investigated as response variables. Figure 6.6 summarises the obtained results for Intersection K3. Qualitative statements derived from the figure are expected to carry over to other intersections in oversaturated conditions.



(a) Median of measured delays

(b) Standard deviation of measured delays

Figure 6.6: Influence of the simulated duration on measured delays for a signal plan in oversaturated conditions (based on 100 replications)

Figure 6.6a depicts the median delay for the different simulated durations. In contrast to the undersaturated case (see Figure 6.5a), the median of the delay measurements increases with the length of the simulation period in oversaturated conditions. While the delays would grow infinitely large in reality, they are bounded due to the network size (that limits the queue lengths) in the simulations. The increase can be observed for every investigated warm-up duration, but is strongest for the tests without a warm-up period. Without warm-up, queues are still building up during the simulation which lowers the average delay determined for the simulation period. Therefore, the use of a warm-up period is recommended in case that a correct delay estimate is required.

Compared to the undersaturated scenario, the standard deviation of the delay measurements is significantly higher for the oversaturated case (compare Figures 6.5b and 6.6b). However, the effect of extending the simulation period is similar in both cases: The simulation period should be selected sufficiently large to benefit from the initially strongly reduced deviations, but overly long simulations should be avoided unless reduced random noise is much more important than execution time.

Figure 6.6b also depicts the effect of a warm-up period on the standard deviation of delays. For the test without warm-up, a low standard

deviation is observed when considering short simulation periods. Due to the missing warm-up, oversaturated conditions cannot built up fully within a short simulation. As a result, delays measured for different replications exhibit a relatively low variability. As the simulation period increases, delays measured for simulations without warm-up approach those measured in the tests with a preceding warm-up (see Figure 6.6a). Therefore, the apparent initial advantage of a missing warm-up period gets lost over time.

Tests with a preceding warm-up period are mainly unaffected by such start-up effects. Best results in terms of a low standard deviation are obtained for a warm-up period of 1800 s.

Summary The conducted experiment has investigated the relation of simulated duration, required execution time, and the resulting standard deviation of delay estimates. Results confirm a linear relation between simulated duration and execution time which was to be expected for a microscopic simulation environment.

With respect to the standard deviation of delay measurements, simulations are found to exhibit a larger variability when flow is close to, or exceeds, the capacity of the simulated signal plan. Longer simulation periods can reduce the standard deviation of delay estimates. However, since the relation is not linear, the simulation period should be sufficiently long to benefit from the initially strongly reduced deviations, but not overly long to save on execution time.

The use of a preceding warm-up period does not have a positive effect for the investigated single node networks when considering signal plans in undersaturated conditions. Neither the delay estimates nor their standard deviation are positively affected by a preceding warm-up. For signal plans in oversaturated conditions, a preceding warm-up period of 1800 s is recommended to obtain correct delay estimates and reduce random noise.

Ranking of signal plans

The previous experiment considered the standard deviation as measure for the variability of delay estimations. To investigate the influence of

the simulated duration on the ranking of candidate signal plans (which is important in case of evolutionary optimisation due to the employed rank-based survivor selection), a second experiment has been conducted.

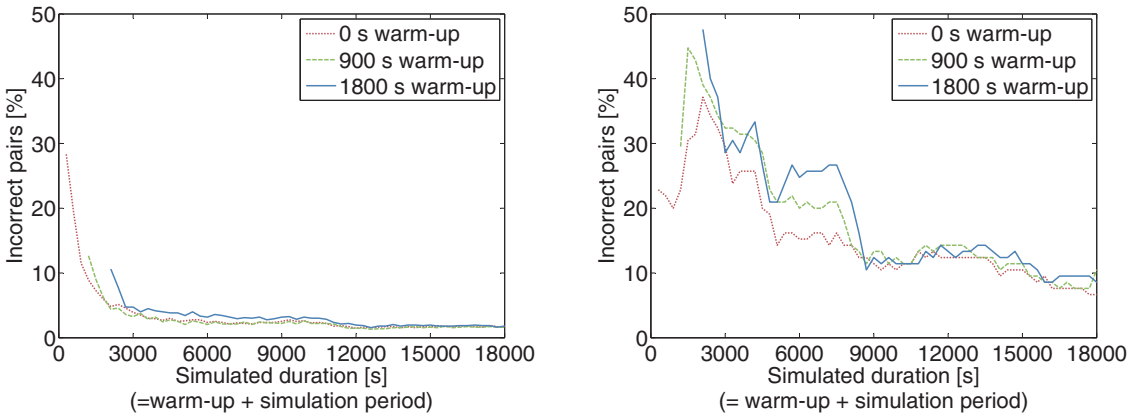
The experiment considers the duration of the warm-up and simulation periods as design factors and uses the same factor levels as before. To investigate the ranking quality, a set of fixed-time signal plans for Intersection K3 has been simulated during the morning peak hour. Based on the vehicular delay estimates obtained after each considered warm-up and simulation period, signal plan rankings are obtained. The quality of these ranking is determined by comparison to a reference ranking.

To obtain the reference ranking, each signal plan in the set has been simulated using 100 replications with different random seeds, a simulated duration of 14 400 s, and a preceding warm-up of 1800 s. The reference ranking has then been computed based on the mean delay estimate obtained for each signal plan.

When comparing the rankings obtained for the different warm-up and simulation periods to the reference ranking, signal plans are investigated pairwise: Each time a ranking assigns signal plan s_1 a better rank than plan s_2 while the reference ranking ranks plan s_2 better than s_1 (or the other way around), this is considered as a wrongly ranked pair. The quality of the ranking is determined by the percentage of wrongly ranked pairs which is the response variable for the conducted experiment.

Figure 6.7 summarises the obtained results for two sets of signal plans. The first set (called *random set*) contains 50 randomly selected signal plans. Like in the beginning of an evolutionary optimisation, the plans exhibit a widely varying fitness in terms of average delay. Therefore, ranking should be a comparatively simple task. The second set (called *converged set*) consists of 15 plans from the last generation of a short evolutionary search. The plans in this set are of a similar quality and should be more difficult to rank.

Ranking random signal plans Figure 6.7a depicts the percentage of incorrectly ranked pairs when evaluating the random set using varying simulated durations. The figure shows that rank errors are quickly reduced as the simulated duration increases. While the initial reduction rate is high, the beneficial effect on the ranking error is nearly lost as



(a) 50 random signal plans

(b) 15 converged signal plans

Figure 6.7: Influence of the simulated duration on the ranking of signal plans

the simulated duration increases further. The findings are consistent with the non-linear relation of simulated duration and standard deviation observed previously (see Figures 6.5b and 6.6b).

With respect to a preceding warm-up period, no beneficial effects can be deduced from the figure although several signal plans in the random set operate in oversaturated conditions during the morning peak. For these plans, a preceding warm-up period is recommended to obtain correct delay estimates (see Figure 6.6). Correct delay estimates are, however, no necessary precondition to obtain a correct ranking, since the ranking considers the relative quality of signal plans, only.

Ranking converged signal plans While signal plans of a widely varying quality can be ranked quite accurately based on short simulations, the situation becomes more difficult for plans that exhibit a similar vehicular delay (like those occurring in the later generations of an evolutionary search). Figure 6.7b depicts the percentage of incorrectly ranked pairs for 15 similar signal plans. All plans operate in undersaturated conditions and have been obtained by an evolutionary optimisation that considered Intersection K3 during the morning peak hour.

As for the random set, an increasing simulated duration leads to a non-linear error rate reduction when ranking the converged set. However, a higher percentage of incorrectly ranked pairs is observed for nearly all considered simulated durations (compare Figures 6.7a and 6.7b). Therefore, significantly longer simulations are required to reach a predefined ranking quality when a set of similar signal plans is considered.

With respect to a preceding warm-up period, no benefit can be deduced from Figure 6.7b. While the lowest error rate for short simulation periods is obtained without a preceding warm-up, the error rates observed for different warm-up periods converge as the simulation period increases. This resembles the influence of a preceding warm-up period on the standard deviation of signal plans in undersaturated conditions (see Figure 6.5b). For a set of converged signal plans, a low variability in the delay estimates helps to improve the ranking.

Approximation-based ranking Instead of using time-consuming stochastic simulations to rank a set of signal plans, rankings can be based on an approximation formula when fixed-time plans are considered. Since approximation-based quality estimates can be computed quickly, they are an interesting alternative to simulations if the resulting ranking turns out to be sufficiently accurate.

To evaluate the quality of approximation-based fitness rankings, the random and converged signal plan sets have been ranked with Webster's delay formula. Since Webster's formula (see Equation 2.3) is only applicable to undersaturated conditions, a degree of saturation $g < 1$ is required for each signal group (otherwise the signal plan cannot handle the traffic demand). Since this requirement cannot be guaranteed especially for signal plans contained in the random set, a degree of saturation $g = 0.99$ has been assumed for oversaturated signal groups.

For the random set, the percentage of incorrectly ranked pairs for the Webster approximation is 22.4 %. Compared to a simulation-based ranking, this error rate is remarkably high. A simulated duration of only 600 s is sufficient to obtain a better ranking quality (see Figure 6.7a). For the converged set, an error rate of 22.9 % has been observed. A lower error rate can be reached using simulation-based rankings, however, a simulated duration of at least 4500 s is required (considering the test without warm-up, see Figure 6.7b).

In contrast to rankings that are based on simulations, the approximation-based ranking produces similar error rates for the converged and random sets. This strange behaviour is due to a weakness of Webster's formula that overestimates the delay when an intersection (or a signal group) is nearly saturated (see Section 2.2.2). For the randomly selected signal plans, this leads to an unduly bad rating for some plans and as a result to an increased number of wrongly ranked pairs. In the course of evolution, these critical candidates are typically discarded from the population such that the converged set (that does not contain oversaturated plans) can be ranked with a relatively high accuracy.

Summary When ranking a set of signal plans based on microscopic simulations, the error rate depends on the simulated duration in a non-linear way. An increasing simulated duration initially reduces the ranking error, but overly long simulations have a limited beneficial effect, only. The optimal duration depends on the similarity of the considered signal plans: Plans of a similar quality have to be simulated for a longer period than plans of a widely varying fitness. In any case, a preceding warm-up does not show a beneficial effect on the ranking quality.

When fixed-time plans are considered, rankings can also be based on Webster's delay approximation formula. The formula can be evaluated quickly, but is applicable to undersaturated conditions, only. Here, the ranking error resembles that of a simulation-based ranking for medium simulated durations.

6.3.2 Repeated simulations

Instead of increasing the simulation period to reduce the variability of simulation results and improve the signal plan ranking, it might be beneficial to calculate the average result of several shorter simulations to obtain a solution's fitness. However, the overhead in execution time for the additional starts of the simulation environment and – if applicable – the additional warm-up periods have to be considered.

Table 6.1 summarises the results of an experiment that considers the number of repeated simulations and their simulation period as design factors. The standard deviation of the vehicular delay measurements and

the time required for executing the simulations serve as response variables. Tests have been performed for undersaturated and oversaturated traffic conditions using the morning plan of Intersection K3. The table lists the response variables for 100 fitness evaluations, where an evaluation is based on

- a single simulation with a simulation period of 14 400 s,
- the average result of two simulations of 7200 s, or
- the average result of four simulations of 3600 s,

respectively. No warm-up period has been used in the experiment.

Table 6.1: Influence of repeated simulations on standard deviation and execution time (uncorrected overhead)

	Undersaturated demand		Oversaturated demand	
	Std. deviation	Exec. time	Std. deviation	Exec. time
1 × 14 400 s	0.1560	297 s	1.0338	516 s
2 × 7200 s	0.1478 (−0.0082)	330 s (+33 s)	1.4775 (+0.4437)	533 s (+17 s)
4 × 3600 s	0.1458 (−0.0102)	394 s (+97 s)	1.5964 (+0.5626)	575 s (+59 s)

Considering the standard deviation of obtained delay estimations, Table 6.1 indicates a slightly beneficial effect of repeated simulations in the undersaturated scenario. However, this improvement comes at the cost of increased execution times since the overhead for the additional starts of the simulation environment has not been considered. For the oversaturated scenario, repeated simulations result in an increased standard deviation. Therefore, a single simulation run with a long simulation period is to be preferred over several shorter runs in oversaturated conditions. The single run combines the lowest standard deviation and the shortest execution time.

To check whether repeated simulations remain beneficial in undersaturated conditions when the overhead in execution time is considered, a follow-up experiment has been conducted. The simulation period of the repeated simulations has been reduced to 6475 s (instead of 7200 s) and

to 2350 s (instead of 3600 s) such that the resulting execution time is approximately the same for all factor combinations. The results of this second experiment are summarised in Table 6.2.

Table 6.2: Influence of repeated simulations on standard deviation and execution time (corrected overhead)

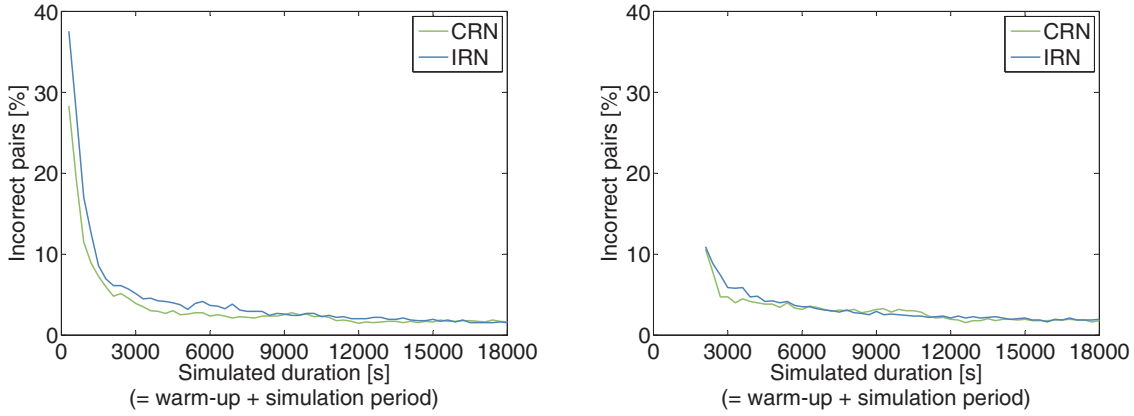
	Undersaturated demand	
	Std. deviation	Execution time
1 × 14 400 s	0.1560	297 s
2 × 6475 s	0.1580 (+0.0020)	307 s (+10 s)
4 × 2350 s	0.1948 (+0.0388)	302 s (+5 s)

With the corrected simulation periods, the slight benefit of repeated simulations that has initially been observed is lost. Despite the undersaturated traffic conditions, the standard deviation of the obtained delay estimates is slightly increased when repeated simulations with short simulation periods are performed. Therefore, it is recommended to use a single but longer simulation run for the evaluation of candidate solutions in both undersaturated and oversaturated conditions. These observations comply to the recommendation of Kesur [104].

6.3.3 Simulation seed

When ranking signal plans based on stochastic simulations, the literature reviewed in Section 3.1.4 recommends to configure the simulator with the same random seed to reduce the variability of simulation results: The random seed determines all random choices made in the course of a stochastic traffic simulation, i. e., it defines the generation and movement of all simulated vehicles (see Section 2.5.1 for details). The aim of using *common random numbers* (CRN) when ranking signal plans is to perform the tests under identical conditions, i. e., the signalisation should be subjected to the same traffic flow pattern. The following experiments investigate the benefits of CRN for different simulated durations and for signal plans operating in under- and oversaturated conditions.

Common random numbers and simulated duration To test whether CRN are beneficial when ranking signal plans based on AIMSUN simulations of varying length, 50 randomly selected fixed-time plans for Intersection K3 have been ranked based on their vehicular delay during the morning peak hour: All signal plans have been simulated for different random seeds to obtain a ranking based on *independent random numbers* (IRN), while all simulations for the CRN-based ranking use the same seed. Figure 6.8 depicts the percentage of wrongly ranked signal plans for both rankings and varying simulated durations.



(a) No warm-up (b) 1800 s warm-up

Figure 6.8: Influence of random seeds on the ranking of signal plans

The figure indicates a reduced error rate for rankings based on CRN especially when the simulated duration is short. The effect can be observed for the test without warm-up, but is also present with a preceding warm-up period. For longer simulated durations, the benefit of CRN is lost.

Common random numbers in under- and oversaturated conditions To investigate the benefit of CRN in more detail, two signal plans have been simulated using a fixed set of 100 random seeds. Afterwards, the plans have been ranked based on the resulting delay estimations:

- CRN-based rankings have been obtained for the 100 replications that use the same random seed.

- IRN-based rankings have been obtained by considering the 9900 combinations of different random seeds.

For IRN and CRN rankings, the percentage of wrong classifications has been determined under the assumption that the correct ranking of both signal plans can be obtained by comparing the mean vehicular delays of all simulations.

In the conducted experiment, the morning plan of Intersection K3 and a modified version of this signal plan have been compared based on AIMSUN simulations. Both signal plans have been simulated for the same set of 100 random seeds. Simulations ran without warm-up for a simulated period of 7200 s. Undersaturated conditions (at 11 a. m.) and oversaturated conditions (during the morning peak) have been investigated separately. Table 6.3 summarises the obtained mean delays that reflect the true ranking of both signal plans. Results show that the morning plan performs slightly better than its modification for under- and oversaturated conditions.

Table 6.3: Mean delay estimates (and their standard deviation) for ranked signal plans

	Undersaturated conditions	Oversaturated conditions
K3 morning plan	19.1 s (0.21 s)	33.3 s (1.58 s)
K3 modified plan	19.5 s (0.22 s)	34.3 s (1.24 s)

Based on the signal plans' mean delays, the percentage of wrong classifications has been determined when ranking the plans based on CRN and IRN, respectively. The results are summarised in Table 6.4.

Table 6.4: Influence of random seeds on the error rate of signal plans rankings

	Undersaturated conditions	Oversaturated conditions
CRN	6.0 % (6/100)	8.0 % (8/100)
IRN	9.2 % (907/9900)	29.2 % (2895/9900)

For AIMSUN simulations, the use of CRN is beneficial when comparing signal plans: The advantage is relatively limited in undersaturated conditions where the vehicular delays exhibit a small standard deviation, only. However, a remarkable benefit can be obtained in oversaturated conditions, where the use of CRN is strongly recommended. The above results are consistent with the findings of Rathi [152] and Kesur [104] who investigated the benefit of CRN for the TRAF-NETSIM [153] and MSTRANS [103] simulators (see Section 3.1.4).

Summary When ranking signal plans based on AIMSUN simulations, CRN can reduce the ranking error. The use CRN is most beneficial when simulation results suffer from a high variability, i. e., when the simulated duration is limited or when the considered signal plans are operating in oversaturated conditions.

6.4 Evolutionary Algorithm

To successfully evolve near-optimal signal plans with an EA, one has to identify a suitable configuration for the EA's design factors. In a sensitivity study, factors that largely influence the outcome of the evolutionary search have to be identified, before appropriate levels for these factors can be determined.

The following sections identify suitable EA configurations for varying fitness landscapes: Section 6.4.1 focuses on the deterministic landscape that results from the use of Webster's approximation, while EA configurations for simulation-based (and therefore noisy) fitness functions are in the focus of Section 6.4.2. Section 6.4.3 investigates advanced aspects related to simulation-based optimisations that include the handling of random seeds and simulated durations throughout the search. The findings for approximation- and simulation-based landscapes are summarised in Section 6.4.4.

6.4.1 Approximation-based fitness landscape

A first sensitivity study considers optimisations that rely on Webster's approximation formula (see Section 2.2.2) to evaluate the quality of

fixed-time signal plans. Webster's formula gives deterministic results and can be computed quickly such that an extensive evolutionary search is feasible.

The study's goal is to determine a suitable EA configuration that allows to evolve near-optimal signal timings. The basic EA framework is given by the self-adaptive Evolution Strategy that is defined in Algorithm 5.1. The EA's parent and offspring population size (that determine the selection pressure) and the selection strategy are considered as design factors. Using a factorial experimental design, the factor level combinations given in Table 6.5 are investigated. Each combination is tested in 50 replicated optimisation runs to account for the stochastic character of the search.

Table 6.5: EA configurations for the approximation-based fitness landscape

Design factor	Considered levels
Start population	Random
Population size	
(μ, λ) selection	$\mu = 16,\ \lambda \in \{48, 64, 80, 96, 112\}$
$(\mu + \lambda)$ selection	$\mu = 16,\ \lambda \in \{24, 32, 48, 64, 80\}$
Selection strategy	(μ, λ), $(\mu + \lambda)$
Mutation operator	Gaussian perturbation, self-adaptive step size
Crossover operator	Discrete crossover
Evaluation function	Webster approximation
Stopping criterion	Max. 1552 evaluations

Signal plan optimisations are conducted for Intersection K3, use the average traffic demand of the morning peak hour (7:30 a. m. till 8:30 a. m.), and aim at minimising the average vehicular delay. A reference signal plan for this demand has been obtained by enumerating all feasible signal timings. The reference plan exhibits an average vehicular delay of 25.98 s which is considered to be optimal.

Figure 6.9 visualises the vehicular delays for the investigated factor combinations in form of a boxplot. Each factor combination is represented by one box. On each box, the central mark indicates the median of the measured delays, the edges of the box are the 25th and 75th percentiles. Whiskers extend to the most extreme measurements that

are not considered outliers. A measurement is an outlier if it is larger than $q_3 + 1.5 \cdot (q_3 - q_1)$ or smaller than $q_1 - 1.5 \cdot (q_3 - q_1)$, where q_1 and q_3 are the 25th and 75th percentiles, respectively. Outliers are plotted individually.

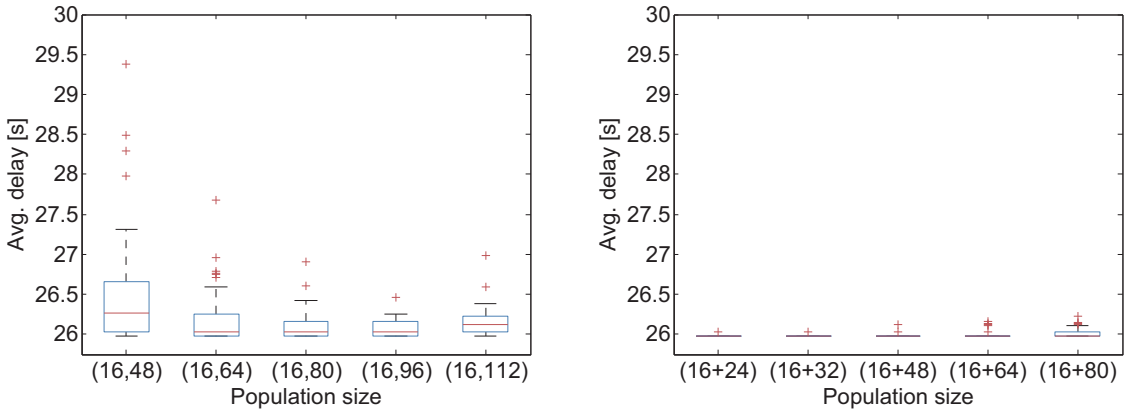
Figure 6.9 depicts the vehicular delays of the best evolved signal plans. For (μ, λ) strategies, these signal plans might not be part of the final generation. The following conclusions can be drawn from the figure:

Population size The most suitable size of the offspring population depends on the selection strategy. When (μ, λ) selection is applied, surviving individuals are selected among the offspring, only. In consequence, the offspring population needs to contain relatively fit individuals that can guide the search. This becomes more likely with an increasing population size. However, a large offspring population limits the number generations that can be evolved before reaching the stopping criterion. A population size of $\lambda = 96$ (that allows to evolve 16 generations) constitutes the best investigated trade-off (see Figure 6.9a). The setting corresponds to a μ/λ ratio of 1/6 which is within the literature recommendations for the selection pressure in (μ, λ) strategies (see Section 5.4.2).

For $(\mu + \lambda)$ selection, the influence of the offspring population is limited (see Figure 6.9b). Near-optimal signal plans are evolved for all investigated population sizes. As parent and offspring populations are considered during survivor selection, the fitness of the parent population cannot decrease throughout the search. However, smaller offspring populations allow to evolve more generations before the stopping criterion is reached. In consequence, population sizes of $\lambda \leq 32$ provide the most stable results.

Selection strategy Considering the different selection strategies, $(\mu + \lambda)$ selection is recommended. None of the investigated factor combinations for (μ, λ) selection can outperform the best $(\mu + \lambda)$ strategies.

In conclusion, signal plans for the deterministic fitness landscape can be optimised successfully by a self-adaptive Evolution Strategy. Based on the results of the sensitivity study, a $(\mu + \lambda)$ selection scheme is recommended as it allows to evolve (near-)optimal signal timings in combination with various population sizes.



(a) (μ, λ) selection

(b) $(\mu + \lambda)$ selection

Figure 6.9: Comparison of EA configurations for the approximation-based landscape

6.4.2 Simulation-based fitness landscape – Basic configuration

Sometimes, e. g., in the traffic-actuated case, signal plans have to be evaluated by simulation. Although traffic simulations are widely applicable due to their flexibility, they are also time-consuming and noisy (see Section 6.3). Therefore, EA configurations suitable for simulation-based landscapes are investigated in a separate sensitivity study.

The study's experimental setup is similar to the approximation-based case: The self-adaptive Evolution Strategy (Algorithm 5.1) needs to be configured to evolve delay-minimal signal plans. The parent and offspring population size and the selection strategy are considered to be design factors. In contrast to the previous study, EA runs are limited to a maximum of 400 fitness evaluations to attribute for the large time requirements of traffic simulations. In consequence, the parent and offspring populations are chosen smaller than in the previous study. The investigated factor levels are summarised in Table 6.6.

To account for the stochastic character of the evolutionary search, each factor combination is evaluated in 50 replicated EA runs. Runs minimise the average vehicular delay at Intersection K3 for the average demand of the morning peak hour (7:30 a. m. till 8:30 a. m.) and obtain delay

Table 6.6: EA configurations for the simulation-based fitness landscape

Design factor	Considered levels
Start population	Random
Population size	
(μ, λ) selection	$\mu = 8,\ \lambda \in \{32, 40, 48\}$
$(\mu + \lambda)$ selection	$\mu = 8,\ \lambda \in \{16, 24, 32\};\ \mu = 16,\ \lambda \in \{24, 32, 40\}$
Selection strategy	(μ, λ), $(\mu + \lambda)$
Mutation operator	Gaussian perturbation, self-adaptive step size
Crossover operator	Discrete crossover
Evaluation function	AIMSUN simulations
Stopping criterion	Max. 400 evaluations

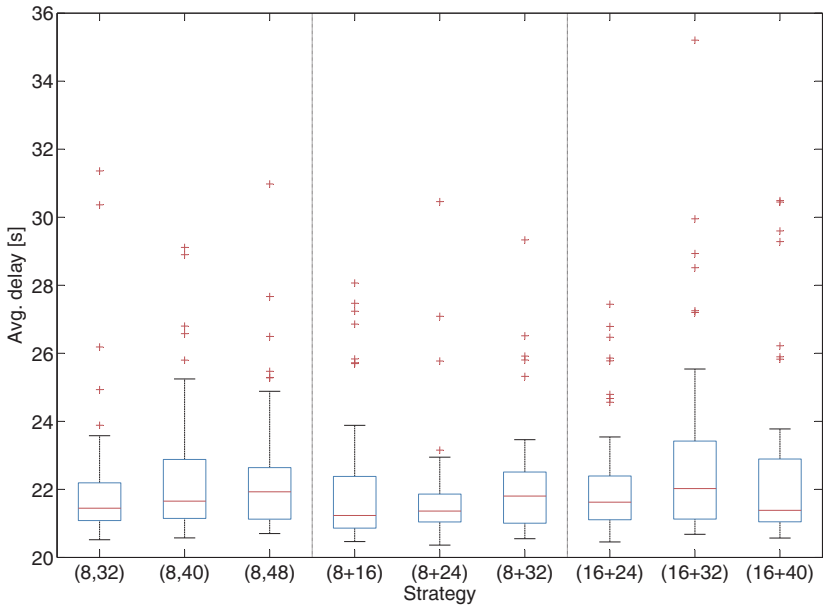
estimates from AIMSUN simulations that use a simulation period of 3600 s without a preceding warm-up. After each run, the best evolved signal plan is simulated for ten different random seeds and a simulated duration of 14 000 s to determine its true average delay which serves as response variable.

Figure 6.10 summarises the obtained delays in a boxplot and lists the 25th, 50th (median), and 75th percentiles for the different strategies (best values are highlighted by bold font). Due to the different fitness landscape, the depicted delay values are not comparable to Figure 6.9. In contrast to the approximation-based landscape, a delay-minimal signal plan is unknown here. From Figure 6.10, the following conclusions can be drawn for simulation-based signal plan optimisations:

Population size The most suitable population sizes depend on the selection strategy. For (μ, λ) selection, the best signal plans are evolved for $\lambda = 32$. The literature recommends larger offspring populations, but in the investigated scenario, the number of evaluations is strictly limited. In consequence, larger offspring populations overly reduce the number of evolvable generations.

For $(\mu + \lambda)$ selection, the combination of small parent and offspring populations (i. e., $\mu = 8$ and $\lambda \in \{16, 24\}$) gives the best results. The lowest 25th, 50th (median), and 75th percentiles are obtained for these factor combinations.

Selection strategy In contrast to the approximation-based landscape, (μ, λ) selection is competitive in the simulation-based case. As (μ, λ) strategies discard the parents during survivor selection, they are less susceptive to random noise.



Strategy → / ↓ Percentile	(8,32)	(8,40)	(8,48)
25th	21.08	21.15	21.13
50th	21.45	21.65	21.93
75th	22.19	22.88	22.64

Strategy → / ↓ Percentile	(8+16)	(8+24)	(8+32)	(16+24)	(16+32)	(16+40)
25th	**20.86**	21.04	21.01	21.11	21.13	21.05
50th	**21.23**	21.36	21.80	21.62	22.02	21.38
75th	22.38	**21.86**	22.51	22.39	23.42	22.89

Figure 6.10: Comparison of EA configurations for the simulation-based landscape

Based on these findings, a self-adaptive Evolution Strategy with $(\mu+\lambda)$ selection and population sizes $\mu = 8$ and $\lambda = 24$ is recommended as basic EA configuration for the simulation-based fitness landscape considering

the given run-time restrictions. Now, the configuration will be further refined.

6.4.3 Simulation-based fitness landscape – Advanced configuration

While the previous section identified a suitable configuration for the EA, specific aspects related to the simulation-based evaluation of signal plans have not been addressed:

Simulated duration Traffic simulations are time-consuming. While the required execution time for a simulation linearly increases with the simulated duration, longer simulated durations decrease the random seed's influence (see Section 6.3.1). This leads to a trade-off: Shorter simulations allow for a more extensive search within a given time budget, but they come at the cost of increased random noise. Therefore, the optimal simulated duration depends on the amount of noise that is acceptable for the EA. This amount might vary as the population improves and converges throughout the search.

Simulation seed The outcome of a microscopic simulation is affected by the simulator's random seed. Section 6.3.3 has shown that signal plan rankings are less error-prone when they are based on the same simulation seed (CRN-based ranking). A simple way to take advantage of this observation in an evolutionary search is to keep the simulation seed fixed throughout the optimisation. A fixed seed allows for fair signal plan comparisons, but might bias the search in favour of the seed-specific experimental conditions. More robust signal plans might be evolved when the simulation seed is varied with the generations. To benefit from CRN in the case of varied seeds, time-consuming reevaluations that reduce the extent of the search can be required, though.

Border case signal plans The EA's rank-based survivor selection deterministically selects the μ best signal plans for the next generation. Errors in the signal plan ranking affect the evolutionary process only in case that a wrongly ranked pair has ranks $r_1 > \mu$ and

$r_2 \leq \mu$. Therefore, it might be beneficial to reevaluate signal plans with a rank close to μ with a longer simulated duration. Assuming a limited optimisation time budget, reevaluations reduce the extent of the search, though.

The different strategies for handling simulation-based fitness evaluations within the EA are investigated in the following experiments.

Simulated duration

When conducting a simulation-based signal plan optimisation, traffic simulations determine the fitness landscape for the evolutionary search and are responsible for the largest share of the EA's run-time. Unfortunately, short simulations and low levels of noise cannot be obtained simultaneously (see Section 6.3.1). To reduce random noise in fitness evaluations, their simulated durations have to be increased which in turn increases the required execution time. As an EA benefits from both, numerous and accurate fitness estimates, a limited time budget poses questions regarding the efficient use of simulations:

1. Should the time available for optimisation be spent on relatively few, but accurate simulations, or is it better to perform a larger number of less accurate simulations?

2. Are there stages within an evolutionary search during which an EA is especially susceptive to noise?

To address both questions, different strategies for distributing a simulation time budget throughout an evolutionary search have been compared. Comparisons are based on the outcome of 50 optimisation runs that consider the morning peak demand of Intersection K3 and aim at finding a signal plan that minimises the average vehicular delay. After each optimisation, the best signal plan found by the EA has been simulated for ten different random seeds and a simulated duration of 14 000 s per seed to determine its true average delay.

Based on the findings of Section 6.4.2, optimisations use population sizes of $\mu = 8$ and $\lambda = 24$ and $(\mu + \lambda)$ selection. Fitness evaluations use a fixed seed throughout the search. To allow for a fair comparison of signal

plans, surviving plans are reevaluated after a change in the simulated duration. Simulations are performed with no warm-up, as a preceding warm-up period does not reduce the ranking error (see Section 6.3.1).

The time budget for traffic simulations has been set to 1 440 000 simulated seconds per optimisation. This budget can be freely allocated throughout the optimisation, resulting in a varying number of generations available for different allocation strategies. Strictly, the budget should be based on execution time instead of simulated duration as the start of a simulation produces a time overhead of approximately 0.3 s per simulation (see Figure 6.4). However, balancing the execution times of EA runs is difficult as the time requirements of a simulation depend on the number of simulated vehicles which (for a fixed traffic demand) depends on the simulated signal plan. Therefore, execution times vary slightly, but unpredictably, among EA runs.

Table 6.7 lists the distribution strategies that have been investigated in the experiment. Figure 6.11 additionally depicts graphical illustrations. The investigated strategies are partly based on previous findings by Branke [21], who argues that an EA is especially susceptive to noise in the beginning and at the end of a run: In the beginning, the initially diverse population quickly converges towards a region of the search space that looks promising. In the end, the nearly converged population needs to reach a (local) optimum of the fitness function. High levels of noise caused by short simulated durations can misguide the EA in both phases, but the end of the search is considered to be especially critical. The investigated strategies can be subdivided into the following groups:

Constant (C) strategies The first group of strategies (depicted in Figure 6.11a) applies a constant simulated duration throughout the search. The strategies implement different trade-offs among simulated duration and noise.

Final generation (F) strategies Based on Branke's findings, strategies in this group support the correct selection of the optimisation's end result. To obtain highly accurate fitness estimates in the final generation, this generation is evaluated with a simulated duration of 10 800 s. For all previous generations, the simulated duration is kept constant (see Figure 6.11b).

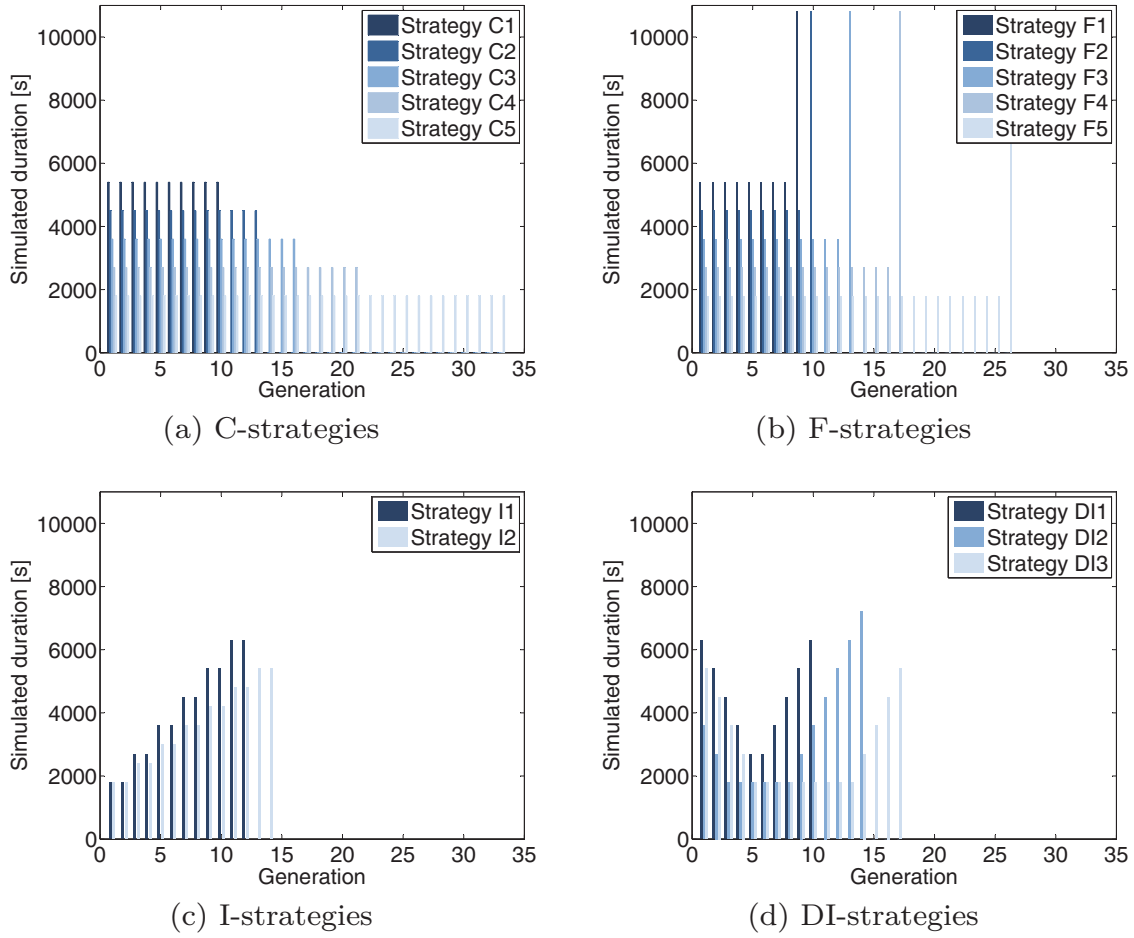
Increasing (I) strategies For the third group of strategies, the simulated duration is linearly increasing. As the population converges, the increased simulated duration should help to discriminate signal plans of similar quality. The strategies in this group are depicted in Figure 6.11c.

Decreasing-increasing (DI) strategies The fourth group of strategies reduces the random noise in the beginning and in the end of the search. The strategies apply a linear decrease in the simulated duration that is followed by a linear increase (see Figure 6.11d).

Table 6.7: Strategies for distributing the simulated duration on generations

Strategy	Generational distribution	Eval.	Gen.	Simulated duration (total)
C1	$10 \times 5400\,\mathrm{s}$	248	10	1 339 200 s
C2	$13 \times 4500\,\mathrm{s}$	320	13	1 440 000 s
C3	$16 \times 3600\,\mathrm{s}$	392	16	1 411 200 s
C4	$21 \times 2700\,\mathrm{s}$	512	21	1 382 400 s
C5	$33 \times 1800\,\mathrm{s}$	800	33	1 440 000 s
F1	$8 \times 5400\,\mathrm{s} + 1 \times 10\,800\,\mathrm{s}$	232	9	1 425 600 s
F2	$9 \times 4500\,\mathrm{s} + 1 \times 10\,800\,\mathrm{s}$	256	10	1 353 600 s
F3	$12 \times 3600\,\mathrm{s} + 1 \times 10\,800\,\mathrm{s}$	328	13	1 411 200 s
F4	$16 \times 2700\,\mathrm{s} + 1 \times 10\,800\,\mathrm{s}$	424	17	1 404 000 s
F5	$25 \times 1800\,\mathrm{s} + 1 \times 10\,800\,\mathrm{s}$	640	26	1 440 000 s
I1	1800 s–6300 s (900 s/2 gen.)	336	12	1 360 800 s
I2	1800 s–5400 s (600 s/2 gen.)	448	14	1 411 200 s
DI1	6300 s–$2 \times 2700\,\mathrm{s}$–6300 s	312	10	1 418 400 s
DI2	3600 s–$6 \times 1800\,\mathrm{s}$–7200 s	408	14	1 425 600 s
DI3	5400 s–$9 \times 1800\,\mathrm{s}$–5400 s	480	17	1 440 000 s

Figure 6.12 depicts a boxplot of the obtained results and shows the 25th, 50th (median), and 75th percentiles for the different distribution strategies. For the C-strategies, obtained deviations in the signal plan quality are relatively large: An overly long simulated duration spends the simulation time budget on too few fitness evaluations which leads to a premature termination of the search (Strategy C1). Shorter simulated



(a) C-strategies

(b) F-strategies

(c) I-strategies

(d) DI-strategies

Figure 6.11: Strategies for distributing the simulated duration on generations

durations allow for a more extensive search by treating evaluation quality for quantity. However, the increasingly noisy evaluations hinder the correct ranking of signal plans in general and the selection of the best signal plan in particular (Strategies C4 and C5). In consequence, the best constant strategies balance simulated duration and noise (Strategies C2 and C3). Nevertheless, constant strategies are not recommended for budgeted optimisations due to their large deviations.

Deviations are reduced for the F-strategies that increase the simulated duration for the final generation of a run. The temporary increase reduces the extent of the search only slightly, but allows for a more accurate

selection of the best signal plan. For Strategies F4 and F5, this leads to a remarkable improvement of the delay median and the observed deviations. Strategy F4 constitutes the best overall strategy that exhibits the lowest 25th, 50th (median), and 75th percentiles (indicated by bold font). Strategies F1 to F3 perform similar to their constant counterparts. While F1 (like C1) suffers from a premature termination of the search, Strategy F2 and F3 cannot outperform their good constant counterparts.

Neither the I- nor the DI-strategies can compete with the best F-strategies. The varying simulated durations consume portions of the simulation time budget that are better spent on a more extensive search. Additionally, changes in simulated duration require the reevaluation of surviving parent signal plans. The reevaluation does not support exploration, but is necessary to ensure a reasonable signal plan ranking within a generation (see Section 6.3.1).

In summary, the most successful strategies combine short simulations throughout their search with an increased simulated duration in the final generation of a run. Short simulations allow for an extensive search despite the given budget limitations, while accurate evaluations in the final generation support the selection of the best signal plan. The importance of the final generation(s) has also been observed by Branke. A benefit of additionally supporting the initial stages of a search (as suggested in [21]) could not be verified, though. The reason is in the investigated selection schemes: In contrast to the overlapping $(\mu + \lambda)$ selection strategy investigated here, Branke considers a non-overlapping generational approach without elitism that does not require the reevaluation of surviving solutions after a change in the estimation accuracy. As reevaluations are costly and limit the extent of the search, distribution strategies that reduce changes in estimation accuracy are favoured for overlapping selection strategies.

Simulation seed

Besides simulated duration, the handling of simulation seeds constitutes a second aspect that needs to be considered for simulation-based fitness evaluations. The simulation seed determines the simulator's random decisions and affects a simulation's outcome. In the following, different seed handling strategies are investigated:

Fixed simulation seed A promising and yet simple strategy uses the same simulation seed throughout the evolutionary search. Due to the fixed seed, signal plans are always compared in a CRN-based ranking that has been found to reduce ranking errors (see Section 6.3.3). However, a fixed seed might bias the search towards the seed-specific vehicle arrival pattern which might result in less robust signal plans.

Varying simulation seeds To avoid a seed-specific bias, the simulation seed can be varied every i generations. Several strategies for treating the fitness of surviving signal plans after a seed change are promising:

Keep fitness (KF) strategies This group of strategies evaluates a signal plan once at creation and keeps its fitness unchanged throughout evolution. Costly reevaluations are thereby avoided, but rankings are based on different seeds if the ranked plans have been created in different generations.

Discard fitness (DF) strategies The second group of strategies reevaluates the fitness of a surviving signal plan after a change of the simulation seed. The plan's previous fitness is discarded. This ensures a CRN-based ranking, but requires the reevaluation of surviving signal plans.

Average fitness (AF) strategies Like in the previous group of strategies, signal plans are reevaluated after a change of the simulation seed. The difference is that a plan's previous fitness is not discarded, but used to compute an averaged fitness for the signal plan. An averaged fitness improves the confidence in the plan's quality estimate, but does not allow for CRN-based rankings.

The different seed handling strategies are compared based on the outcome of 50 optimisation runs. As previously, the runs consider the morning peak demand of Intersection K3 and aim at finding a signal plan that minimises the average vehicular delay. After each optimisation, the best signal plan found by the EA is simulated for ten different random seeds and a simulated duration of 14 000 s per seed to obtain an accurate estimation of its average delay that is independent of the simulation seed.

As in the previous experiment, optimisations apply the Evolution Strategy introduced in Algorithm 5.1 with $(\mu+\lambda)$ selection and population size $\mu = 8$ and $\lambda = 24$. Simulations are performed without a preceding warm-up period, using a constant simulated duration of 2700 s throughout the search and of 10 800 s for the final generation. This corresponds to the most successful distribution strategy identified in the previous experiment (Strategy F4).

Table 6.8 lists the seed handling strategies that have been considered in the experiment. Numbers attached to the strategy names indicate the frequency of seed changes: When applying, e. g., the KF2-strategy, the simulation seed is changed every second generation. To allow for a fair comparison, the strategies have to keep a simulation time budget of 1 440 000 s in total. For the DF- and AF-strategies that reevaluate surviving signal plans, the number of generations has been reduced to keep the given budget.

Table 6.8: Strategies for handling the random seed

Strategy	Simulated duration	Eval.	Gen.	Simulated duration (total)
FIXED	$16 \times 2700\,\text{s} + 1 \times 10\,800\,\text{s}$	424	17	1 404 000 s
KF1	$16 \times 2700\,\text{s} + 1 \times 10\,800\,\text{s}$	424	17	1 404 000 s
KF2	$16 \times 2700\,\text{s} + 1 \times 10\,800\,\text{s}$	424	17	1 404 000 s
KF4	$16 \times 2700\,\text{s} + 1 \times 10\,800\,\text{s}$	424	17	1 404 000 s
DF1	$12 \times 2700\,\text{s} + 1 \times 10\,800\,\text{s}$	416	13	1 382 400 s
DF2	$14 \times 2700\,\text{s} + 1 \times 10\,800\,\text{s}$	424	15	1 404 000 s
DF4	$15 \times 2700\,\text{s} + 1 \times 10\,800\,\text{s}$	424	16	1 404 000 s
AF1	$12 \times 2700\,\text{s} + 1 \times 10\,800\,\text{s}$	416	13	1 382 400 s
AF2	$14 \times 2700\,\text{s} + 1 \times 10\,800\,\text{s}$	424	15	1 404 000 s
AF4	$15 \times 2700\,\text{s} + 1 \times 10\,800\,\text{s}$	424	16	1 404 000 s

Figure 6.13 depicts a boxplot comparing the different seed handling strategies and gives the obtained 25th, 50th (median), and 75th percentiles for the different strategies (best values are highlighted by bold font). Strategies that vary the simulation seed throughout their search do not clearly outperform the fixed strategy that combines low delays and small deviations. It can be concluded that a fixed seed does not introduce a

bias that negatively affects the robustness of evolved signal plans. The observation is consistent with earlier findings of Rathi [152], Sanchez et al. [164], and Kesur [104] who investigate seed handling strategies for different simulators (see Section 3.1.4).

Consequently, strategies that change the simulation seed throughout their search (i. e., KF-, DF-, and AF-strategies) exhibit a trend towards lower delays and smaller deviations when the change frequency is reduced. Among the strategies that reevaluate surviving signal plans (i. e., DF- and AF-strategies), averaging the survivors' quality estimates gives (slightly) better results than discarding their previous values. Among the strategies that do not apply a survivor reevaluation (i. e., FIXED- and KF-strategies), the fixed strategy benefits from a CRN-based signal plan ranking (that is recommended in Section 6.3.3).

In summary, keeping the simulation seed fixed throughout the evolutionary search is the best investigated seed handling strategy. A fixed seed does not introduce a bias in favour of its specific traffic-conditions, does not require costly reevaluations, and allows for a CRN-based signal plan ranking that reduces the ranking error.

Border case signal plans

Finally, the EA's rank-based survivor selection has been revisited to investigate whether the reevaluation of border case signal plans (i. e., plans with a rank in $[\mu - i + 1, \mu + i]$ for a small i) can improve the evolutionary search. For border case plans, small fitness deviations due to random noise decide on their survival. Therefore, a reevaluation based on an increased simulated duration can help to better discriminate the border cases. Assuming a fixed simulation time budget, the necessary reevaluations reduce the extent of the evolutionary search, though.

In the following, different reevaluation strategies are compared. As in the previous experiments, comparisons are based on the outcome of 50 optimisation runs that consider the morning peak demand of Intersection K3 and aim at finding a signal plan that minimises the average vehicular delay. After each optimisation, the best signal plan found by the EA is simulated for ten different random seeds and a simulated duration of 14 000 s per seed to determine its true average delay.

Based on previous findings, optimisations are performed with the Evolution Strategy introduced in Algorithm 5.1 using population sizes of $\mu = 8$ and $\lambda = 24$ and $(\mu + \lambda)$ selection. A simulated duration of 2700 s throughout the search is combined with an increased duration of 10 800 s for the final generation (Strategy F4). All simulations are conducted without a preceding warm-up period. The simulation seed has been kept fixed throughout the search.

Table 6.9 lists the investigated reevaluation strategies that keep a simulation time budget of 1 440 000 simulated seconds per optimisation. Strategy BC0 does not reevaluate the border case signal plans and serves as reference. The BCn-t strategies reevaluate n $(= 2i)$ border cases for a simulated duration of t seconds (for $n \in \{2, 4, 6\}$ and $t \in \{3600\,\text{s}, 5400\,\text{s}, 7200\,\text{s}\}$).

Table 6.9: Strategies for reevaluating border cases

Strategy	Simulated duration	Eval.	Gen.	Simulated duration (total)
BC0	$16 \times 2700\,\text{s} + 1 \times 10\,800\,\text{s}$	424	17	1 404 000 s
BC2-3600	$14 \times 2700\,\text{s} + 1 \times 10\,800\,\text{s}$	404	15	1 375 200 s
BC4-3600	$13 \times 2700\,\text{s} + 1 \times 10\,800\,\text{s}$	404	14	1 396 800 s
BC6-3600	$12 \times 2700\,\text{s} + 1 \times 10\,800\,\text{s}$	400	13	1 404 000 s
BC2-5400	$14 \times 2700\,\text{s} + 1 \times 10\,800\,\text{s}$	404	15	1 425 600 s
BC4-5400	$12 \times 2700\,\text{s} + 1 \times 10\,800\,\text{s}$	376	13	1 404 000 s
BC6-5400	$11 \times 2700\,\text{s} + 1 \times 10\,800\,\text{s}$	370	12	1 436 400 s
BC2-7200	$13 \times 2700\,\text{s} + 1 \times 10\,800\,\text{s}$	378	14	1 396 800 s
BC4-7200	$11 \times 2700\,\text{s} + 1 \times 10\,800\,\text{s}$	348	12	1 396 800 s
BC6-7200	$9 \times 2700\,\text{s} + 1 \times 10\,800\,\text{s}$	310	10	1 339 200 s

Figure 6.14 visualises the obtained results in a boxplot. Additionally, the figure gives the 25th, 50th (median), and 75th percentiles for the different strategies (best values are highlighted by bold font). Results indicate that increasing the number n of reevaluations is not beneficial. An increase leads to a raising vehicular delay that can be observed independent of the simulated duration t. As the quality difference of reevaluated plans is expected to grow with n, the benefit of reevaluations for survivor selection is diminished.

Similarly, an increase in the simulated duration t leads to increased delays. Overly long simulated durations are not required to discriminate the reevaluated signal plans, but reduce the extent of the evolutionary search. As the best investigated reevaluation strategies do not outperform the reference strategy, it is recommended to abstain from a reevaluation of border case signal plans.

6.4.4 Summary

In Section 6.4, EA configurations that are suitable for signal plan optimisation have been identified. The investigated scenarios apply either an approximation-based fitness function or rely on traffic simulations to evaluate a signal plan's fitness.

In the approximation-based scenario, Webster's delay formula (see Section 2.2.2) approximates the average vehicular delay for a signal plan. The formula is only applicable for fixed-time plans, but provides deterministic quality estimates and can be evaluated quickly. Near-optimal signal plans have been evolved using the Evolution Strategy introduced in Algorithm 5.1. A recommended configuration applies $(\mu + \lambda)$ selection and runs for 64 generations using a population size of $\mu = 16$ and $\lambda = 24$. Using this configuration, optimal signal timings are evolved in the vast majority of cases (see Section 6.4.1).

A signal plan's fitness can also be estimated from (microscopic) traffic simulations. Simulations can handle various signal controllers and are flexible with respect to the considered quality criteria, but they are time-consuming and provide noisy estimations. Their time requirements limit the extent of an evolutionary search in on-line applications, while random noise negatively affects the survivor selection. Assuming that 400 fitness evaluations with a simulated duration of 3600 s are acceptable, Section 6.4.2 identified $(\mu+\lambda)$ selection with population sizes of $\mu = 8$ and $\lambda = 24$ as suitable basic configuration for simulation-based optimisations.

Using this basic configuration, additional design factors related to simulation-based evaluations have been investigated in Section 6.4.3. These factors include the handling of simulated durations and simulation seeds, as well as the reevaluation of border case signal plans. The simulated duration determines the trade-off among required execution

time and estimation accuracy. A given simulation time budget is best spent by combining short simulations throughout the search and long, but accurate, simulations in the final generation of a run. The most successful configuration uses no warm-up and simulated durations of 2700 s and 10 800 s, respectively. The use of a fixed simulation seed throughout the search constitutes the best seed handling strategy. A fixed seed allows for a CRN-based signal plan ranking and does not require costly reevaluations. Finally, a reevaluation of border cases signal plans (that is intended to improve the survivor selection in the presence of noise) could not lead to significant improvements.

6.5 Learning Classifier System

Besides an EA, the LCS is the most important component of an organic traffic controller as it implements the controller's on-line learning mechanism. Thus, the careful configuration of its parameters can significantly affect the controller's performance.

Table 6.10 lists potential design factors for the LCS. Most of the factors are known from Wilson's XCS (see Section 3.2.1), others have been newly introduced with XCS-T (see Section 5.4.1). To determine a reasonable LCS configuration, a factor screening experiment has been conducted. In the experiment, factors known from XCS are treated as held-constant factors. Their levels have been chosen based on literature recommendations [40] combined with problem-specific knowledge:

Classifier creation The prediction of newly created classifiers is initialised according to the delay estimation obtained from the EA. Therefore, a predefined initial prediction ρ_I is not required. Prediction error and fitness are pessimistically initialised: The initial prediction error $\varepsilon_I = 25\,\text{s}$ exceeds the expected error of the delay estimation by far. The initial fitness (that can vary in $[0, 1]$) is chosen as $F_I = 0.01$.

Classifier deletion The population size has been limited to a maximum of $N = 200$ classifiers. This size allows to cover the traffic demands during the study and leaves room for some special events. Should

Table 6.10: Potential design factors for the LCS

	Potential design factors	Considered levels
XCS	*Classifier creation*	
	Prediction	$\rho_I \leftarrow$ EA estimation
	Prediction error	$\varepsilon_I = 25$
	Fitness	$F_I = 0.01$
	Classifier deletion	
	Population size	$N = 200$
	Min. experience	$\theta_{del} = 20$
	Fraction of mean fitness	$\delta = 0.1$
	Classifier update	
	Learning rate	$\beta = 0.2$
	Accuracy determination	$\alpha = 0.1$, $\varepsilon_0 = 2$, $\nu = 5$
XCS-T	Activation interval	$n_c \in \{1, 2, 3\}$
	Similarity tolerance (per lane)	$w \in \{40, 60, 80\}$

it become necessary, classifiers are deleted with a probability that is proportionate to their action set size estimate. The deletion probability of a classifier is increased if its experience exceeds $\theta_{del} = 20$ and its fitness is below a δ-fraction of the population's mean fitness.

Classifier update Classifier updates apply the learning rate β and the accuracy determination parameters α and ν with their respective standard settings (see Table 6.10), but use a problem-specific error tolerance ε_0: Classifiers with a prediction error below $\varepsilon_0 = 2\,\mathrm{s}$ are considered to be accurate.

Parameters newly introduced with XCS-T (see Section 5.4.1) are considered to be design factors that are investigated in a factor screening experiment:

Activation interval Once XCS-T has selected a signal plan, the plan is kept active for (at least) n_c cycles: While this activation interval should be sufficiently large to evaluate the signal plan's performance based on a reasonably long observation period, it should not be

overly large to allow for a fast reaction to changing demands. In the experimental study, n_c has been set to one, two, or three cycles.

Similarity tolerance When XCS-T encounters an unmatched traffic demand, its discovery component obtains an optimised signal plan from the EA. The optimised plan is stored in a classifier that covers the traffic demand considered during optimisation. The similarity tolerance defines the width of the classifier's interval predicates. A small tolerance results in specific classifiers that match only few demands which leads to large populations and numerous optimisations. A large tolerance, on the other hand, results in overly general classifiers that negatively affect the performance. In the experimental study, similarity tolerances of 40, 60, and 80 vehicles per hour and lane have been tested.

In the following factor screening experiment, the EA supporting the classifier discovery within XCS-T relies on Webster approximations and is configured based on the findings in Section 6.4.1: A self-adaptive Evolution Strategy running for 64 generations and applying $(\mu + \lambda)$ selection for $\mu = 16$ and $\lambda = 24$ provides optimised signal plans to XCS-T.

As in previous experiments, Intersection K3 serves as test case for the factor screening. The intersection is simulated for six hours during the morning period (from 6 a. m. till noon). During this period, the traffic demand remarkably varies (see Figure 6.2a) which allows for a realistic assessment of different configurations. At the beginning of a simulation run, the classifier population of XCS-T is empty.

Different XCS-T configurations are assessed based on the average vehicular delay that is recorded during the simulation. The size of the classifier population and the number of signal plan optimisations are considered as additional criteria. To compensate the stochastic influence of random vehicle arrivals and the optimisation-based classifier discovery, all response variables have been averaged over ten runs.

As the activation interval and the similarity tolerance are considered to be independent factors, their settings have been investigated separately. Section 6.5.1 summarises the results for the activation interval, findings for the similarity tolerance are presented in Section 6.5.2. Section 6.5.3 concludes the discussion by summarising the obtained findings.

6.5.1 Activation interval

The activation interval defines the number of cycles a signal plan is kept active before it is evaluated and eventually replaced by XCS-T. The effect of different activation intervals is illustrated in Figure 6.15. The figure exemplifies XCS-T activations for the morning demand at Intersection K3. Activations are depicted as vertical lines on the time axis. The line colour indicates whether XCS-T left the active signal plan unchanged (grey) or replaced it with a different plan (red). The blue line gives the active signal plan's cycle time. Grey bars in the background indicate the simulated traffic flow (in total for the intersection).

Depending on the number of cycles between XCS-T activations, the average activation interval lasts from 1.2 min ($n_c = 1$, Figure 6.15a) to 3.6 min ($n_c = 3$, Figure 6.15c). However, not every XCS-T activation results in a changed signalisation. On average, signal plans have been changed every 3.3 min ($n_c = 1$), 6 min ($n_c = 2$), or 7.2 min ($n_c = 3$) to adapt the signalisation to changing demands. It should be noted that signal plan changes can be beneficial even though Figure 6.15 shows a constant demand: The depicted traffic flows (grey bars) show the averaged 15 minute demands taken from the traffic census, but do not reflect stochastic fluctuations in the simulation.

The cycle time development depicted in Figure 6.15 shows that signal plan changes performed by XCS-T reflect the variations in demand: While high traffic flows are accompanied by long cycles (that reduce the influence of lost times), XCS-T selects shorter cycles in periods of lower demands (to avoid unused green times). This reasonable relation of traffic flow and cycle length can be observed for all investigated activation intervals, but shorter activation intervals cause smaller (but more frequent) cycle time changes.

Figure 6.15 illustrates the behaviour of XCS-T for different activation intervals, but does not directly allow to draw conclusions on the obtained performance. To this end, vehicular delays obtained for different activation intervals (and a medium similarity tolerance $w = 60$) are summarised in Figure 6.16. The figure depicts the mean vehicular delay averaged over ten replicated runs. Additional error bars indicate the 25th and 75th percentiles.

Averaged over the simulation period, the vehicular delay ranges from 22.9 s (for $n_c = 1$) to 23.8 s (for $n_c = 3$). Best results are obtained for the shortest activation interval that causes frequent LCS activations, thereby allowing for a timely reaction to changing demands. While the delay differs only slightly for activation intervals of one or two cycles (see Figures 6.16a and 6.16b), a decreased performance is observable for three cycle intervals (Figure 6.16c). Therefore, the LCS activation frequency should not exceed two cycles.

The choice of the activation interval does not only influence the obtainable delay reduction, but affects the number of evolutionary optimisations as well as the size of the LCS population. Small classifier populations are desirable for reasons of efficiency as they require less memory for storage and can be handled with less computational power (e. g., when computing the match set). For the same reason, the number of evolutionary optimisations should be kept at a necessary minimum.

Table 6.11 shows the average population sizes and the average number of optimisations for the investigated activation intervals. Results indicate that smaller populations and fewer optimisation are achieved for longer activation intervals that cause a less frequent activation of the LCS and are therefore less sensitive to fluctuations in demand.

Table 6.11: Influence of the activation interval on the classifier population size and the number of optimisations

	Activation interval		
	1 cycle	2 cycles	3 cycles
Population size (avg.)	121.3	93.5	79.3
Optimisations (avg.)	70.0	53.0	43.6

Based on the results in Figure 6.16 and Table 6.11, an activation interval of two cycles (i. e., $n_c = 2$) is recommended. A one-cycle interval reduces the average delay for the simulation period by 0.1 s, only, but requires 32 % more optimisations.

6.5.2 Similarity tolerance

The similarity tolerance is a second parameter that has been newly introduced with XCS-T. It defines the width of interval predicates for new classifiers, thereby affecting the applicability of optimised signal plans obtained from the EA.

Average vehicular delays for different similarity tolerance levels (and an activation interval of $n_c = 2$) are summarised in Figure 6.17. As before, the figure depicts the vehicular delay averaged over ten replicated runs. Additional error bars indicate the 25th and 75th percentiles.

Depending on the chosen similarity tolerance, the vehicular delay averaged over the simulation period is in the range of 22.7 s (for $w = 40$ veh/h) to 23.7 s (for $w = 80$ veh/h). The lowest delay is obtained for the smallest tolerance level. Delays increase with an increasing tolerance, since a small similarity tolerance ensures that classifiers only match traffic demands that are similar to the demand considered during the optimisation of their signal plan. However, small similarity tolerances come at the cost of large classifier populations and require many optimisations (see Table 6.12) which is not desirable for reasons of computational efficiency. Considering the resulting trade-off between computational effort and obtainable delay reduction, a medium similarity tolerance (i. e., $w = 60$ veh/h) is recommended. A smaller tolerance gives a delay that is reduced by 0.3 s, only, but requires about 60 % more optimisations. A larger tolerance, on the other hand, leads to a relatively large increase in delay.

Table 6.12: Influence of the similarity tolerance on the classifier population size and the number of optimisations

	Similarity tolerance		
	40 veh/h	60 veh/h	80 veh/h
Population size (avg.)	158.1	93.5	61.2
Optimisations (avg.)	85.1	53.0	36.1

6.5.3 Summary

The on-line signal plan selection at an organic intersection is performed by XCS-T, a classifier system that implements Level 1 of the controller

(see Section 5.4.1). XCS-T is based on Wilson's XCS, but introduces an activation interval and a similarity tolerance as novel design factors. Since both factors affect the performance and the computational requirements of XCS-T, their configuration has been investigated in a factor screening experiment. The experiment identified an activation interval of $n_c = 2$ cycles and a similarity tolerance of $w = 60$ vehicles per hour and lane as the best trade-off among the obtainable average vehicular delay and the required computational effort.

The investigations on the configuration of XCS-T conclude the sensitivity study of the controller. Combined with the findings on suitable EA configurations in Section 6.4, the controller is well configured. Its performance will be evaluated in the following section.

6.6 Simulation results

To assess the potential benefits of organic signal control, the observer/controller architecture has been tested in a simulation study. The study is conducted for Intersections K3 and K7 located at Hamburg (see Section 6.1). Both intersections are simulated for 13 hours (from 6 a. m. to 7 p. m.) using a workday demand that has been recorded during a traffic census. Their time-dependent field signal plans serve as reference for comparison. To allow for an on-line evaluation and optimisation, it is assumed that both intersections are equipped with additional detectors for measuring the traffic flows.

The observer/controller optimises cycle length and phase durations at its intersection, but keeps phase sequence of the reference signal plans unchanged. By obeying predefined minimum phase durations and operating with unchanged phase transitions, a safe signalisation for all road users (motorised vehicles, cyclists, and pedestrians) is ensured. Unless otherwise stated, the observer/controller is configured according to the findings in Sections 6.4 and 6.5. Experimental setups for approximation-based and simulation-based signal plan optimisations have been investigated.

The assessment of the observer/controller architecture relies on the average vehicular delay as measure of effectiveness. The delay obtained for the simulation period is compared to the delay resulting from the field signal plans. Comparisons consider the average results of ten replicated

simulation sets to account for stochastic influences. Each simulation set has its own random seeds and consists of three consecutive days (labelled Day 1, Day 2, and Day 3). At the beginning of Day 1, the LCS population is empty. For Days 2 and 3, the classifier population learnt on the previous day(s) is available.

In addition to the average vehicular delay, statistics on the development of the LCS population size and the number of optimisations are presented. Results obtained for Intersections K7 and K3 are summarised in Sections 6.6.1 and 6.6.2, respectively. Section 6.6.3 concludes the discussion of the observer/controller architecture.

6.6.1 Results for Intersection K7

The observer/controller architecture has been evaluated at Intersection K7 for two experimental setups. While signal plan optimisations are based on Webster approximations in the first experiment, AIMSUN simulations are used to evaluate the signal plan quality in the second experiment. Results of both setups are summarised in Figures 6.18 and 6.19, respectively. The figures depict the development of the average vehicular delay for the simulation period. Plotted delays are an average of ten replicated simulation sets. Delays of the reference signal plan and the organic signalisation (labelled OTC for *Organic Traffic Control*) are depicted in red and blue colours, respectively. Additional error bars give the 25th and 75th percentiles for the delay measurements.

Webster approximation

When considering the organic signalisation based on Webster approximations, Figure 6.18a shows an improvement over the field signal plans already on Day 1. Delays are reduced over the whole simulation period, the overall reduction is about 13.7 % compared to the reference solution. Considering that the observer/controller had to learn a suitable signal plan mapping from scratch, the reduction indicates powerful on-line learning capabilities. Shifting the signal plan selection and optimisation from design time to run-time turns out to be feasible: The observer/controller can quickly reduce the vehicular delay compared to the reference solution for the low traffic period preceding the morning peak. In contrast to the

field plan that suits high traffic volumes, the evolved signal plans are well adapted to the low demands in this period. The observer/controller performs slightly better than the reference plan during the morning peak, but clearly improves for the rest of the morning. At noon, when the reference plans are switched, the observer/controller performance drops to the reference plan's level, but quickly improves for the remaining afternoon. Especially the peaks observed for the reference plan are handled well.

On Days 2 and 3 (depicted in Figures 6.18b and 6.18c), the observer/controller benefits from the previously learnt mapping and achieves overall delay reductions of 14.3 % and 14.9 %, respectively. Compared to Day 1, the required number of optimisations is significantly reduced (see Table 6.13a) as many observed traffic demands are already covered by the existing mapping. However, fluctuating vehicle arrivals and an updated signalisation result in unmatched demands that cause the creation of additional classifiers and trigger signal plan optimisations. As the classifier population reaches its size limit early on Day 2, the additional classifiers activate the deletion mechanism of XCS-T. This is indented to get rid of (low fitness) classifiers in well-covered niches. By choosing a larger population size limit, the number of signal plan optimisations on Days 2 and 3 could be further reduced.

Table 6.13: Classifier population size and optimisations for Intersection K7

(a) Webster approximation ($N = 200$)

	Day 1	Day 2	Day 3
Population size (avg.)	164.7	200.0	200.0
Optimisations (avg.)	84.7	37.6	37.5

(b) AIMSUN simulation ($N = 300$)

	Day 1	Day 2	Day 3
Population size (avg.)	257.4	300.0	300.0
Optimisations (avg.)	130.9	34.7	29.2

AIMSUN simulation

A second experiment has evaluated the observer/controller for the case that the simulation module relies on traffic simulations. The experimental setup is based on the findings in Sections 6.4.2, 6.4.3, and 6.5 with two exceptions: Firstly, the maximum LCS population size is increased to $N = 300$ classifiers to ensure that the population can hold the classifiers created on Day 1 in their entirety. Secondly, evolutionary optimisations are stopped after twelve generations to avoid a queuing of optimisation tasks.

Average vehicular delays obtained in the experiment are depicted in Figure 6.19. Table 6.13b presents statistics on the development of the classifier population and on the number of required optimisations. In comparison to a Webster-based simulation module, a slightly higher reduction of the vehicular delay has been obtained for AIMSUN simulations (compare Figures 6.18 and 6.19). The improvement can be attributed to a reduced ranking error achieved by simulation (see Section 6.3.1) or might be a side-effect caused by the dual use of the simulator (see Section 5.5.2). However, additional optimisations conducted for the simulation-based setup in combination with the larger population size limit are a more likely cause.

Table 6.13 shows that on Day 1, optimisations are more frequently conducted when AIMSUN simulations replace Webster approximations in the simulation module. As a simulation-based optimisation can require several minutes to finish, XCS-T can be activated while the adaptation module is still running. In this case, it is possible that XCS-T triggers an additional optimisation that would not have been required if the active optimisation's result had already been included in the mapping. The additional optimisations result in a fine-grained situation-mapping that is likely to cause the observed improvements. On Days 2 and 3, the larger population size limit of the simulation-based setup reduces the number of required optimisations to a level similar (Day 2) or even below (Day 3) that observed for the approximation-based setup.

As the two-levelled learning mechanism applied by the observer/controller successfully reduces the vehicular delay for Intersection K7 compared to the field signal plans, a more complex test case is investigated next.

6.6.2 Results for Intersection K3

The second test case investigated in the observer/controller evaluation is Intersection K3. Compared to the three-armed Intersection K7, K3 is a four-armed junction that is equipped with more signal groups and has to serve higher traffic demands (see Section 6.1). Like in the previous experiment, two experimental setups have been investigated. The first setup relies on Webster approximations in the controller's simulation module, the second setup uses AIMSUN simulations.

Webster approximation

Average vehicular delays obtained for the first experimental setup are summarised in Figure 6.20. The figure depicts vehicular delays averaged over the ten replicated simulation sets. Additional error bars indicate the 25th and 75th percentiles for the delay measurements.

Like at the smaller Intersection K7, the observer/controller can reduce the vehicular delay at Intersection K3 for nearly the whole simulation period. Only at the beginning of the morning peak, it performs slightly worse than the reference plan. In this period, the traffic demand rises steeply such that the observer's preprocessor reports flows that lie below the actual demand. The inaccurate situation parameters cause a suboptimal signal plan selection by the controller. To improve the preprocessing, the observer could apply a shorter rolling horizon (compare Figure 5.2) or make use of historical time series.

Despite the suboptimal performance at the beginning of the morning peak, the average vehicular delay at the intersection is reduced by more than 20 % compared to the field plans. On Days 2 and 3, the observer/controller achieves delay reductions similar to those obtained on Day 1 which indicates a good system performance when learning from scratch.

Statistics on the classifier population size and the number of optimisations are summarised in Table 6.14a. Compared to Intersection K7 (see Table 6.13a), additional optimisations are required which, in consequence, lead to a larger classifier population. The surplus of optimisations is due to a larger input space caused by the higher number of signal groups serving motorised traffic. While the classifiers' interval predicates have to

cover ten signal groups at Intersection K3, seven predicates are sufficient for Intersection K7. Nevertheless, the additional optimisations can be handled without problems.

Table 6.14: Classifier population size and optimisations for Intersection K3

(a) Webster approximation ($N = 200$)

	Day 1	Day 2	Day 3
Population size (avg.)	186.8	200.0	200.0
Optimisations (avg.)	99.4	59.8	58.5

(b) AIMSUN simulation ($N = 400$)

	Day 1	Day 2	Day 3
Population size (avg.)	283.2	379.3	400.0
Optimisations (avg.)	145.1	49.8	32.6

AIMSUN simulation

A second experiment has evaluated the observer/controller for the case that the simulation module relies on AIMSUN simulations. The experimental setup is based on the findings in Sections 6.4.2, 6.4.3, and 6.5, but with two exceptions: To ensure that the mapping can hold all classifiers created on Day 1, its maximum size is increased to $N = 400$ classifiers. Furthermore, the queuing of optimisation tasks is reduced by stopping an evolutionary search after twelve generations.

Figure 6.21 depicts the average vehicular delay obtained in the experiment. Like in the previous test, the observer/controller can reduce the vehicular delay at Intersection K3 for nearly the whole simulation period. Despite an increased delay in the beginning of the morning peak due to a slow reaction of the observer, the overall reduction compared to the reference signal plans is about 20 % and thereby similar to the approximation-based setup.

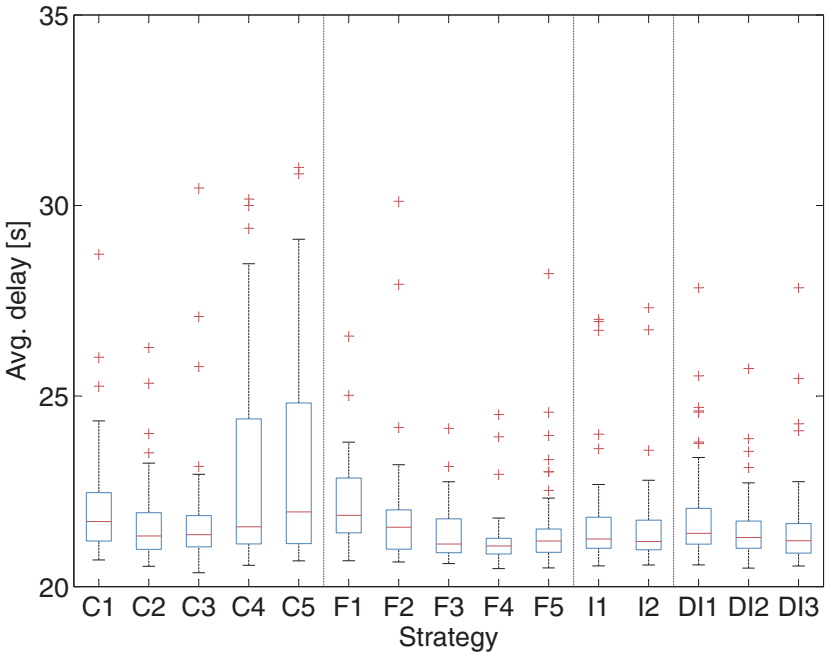
Statistics on the development of the classifier population and on the required optimisations are given in Table 6.14b. In comparison to an

approximation-based simulation module, the simulation-based setup requires an increased number of optimisations and – in consequence – a larger classifier population on Day 1 (compare Tables 6.14a and 6.14b). The phenomenon has already been observed for Intersection K7 and is due to the time requirements of simulation-based optimisations. In contrast, the simulation-based setup requires fewer optimisations on Days 2 and 3. Due to the higher population size limit, traffic demands are more frequently matched by existing classifiers such that optimisations are triggered less often than in the approximation-based scenario.

6.6.3 Summary

In Section 6.6, the observer/controller architecture for traffic control has been evaluated in a simulation study. The study compares the field signal plans of two intersections located at Hamburg, Germany, to an organic signalisation. Comparisons consider typical workday traffic demands and use the vehicular delay as measure of effectiveness.

In the study, significant delay reductions for both test cases are achieved by the observer/controller. Although improved detection capabilities have to be assumed, the obtained results demonstrate the potential of two-levelled learning for signal control, thereby proving that the anticipated shift from design time to run-time configuration is feasible. However, intersections in road networks are typically not operated as stand-alone systems, but have to be coordinated to avoid unnecessary stops. Therefore, self-organising coordination mechanisms for organic intersections are discussed in the following chapter.

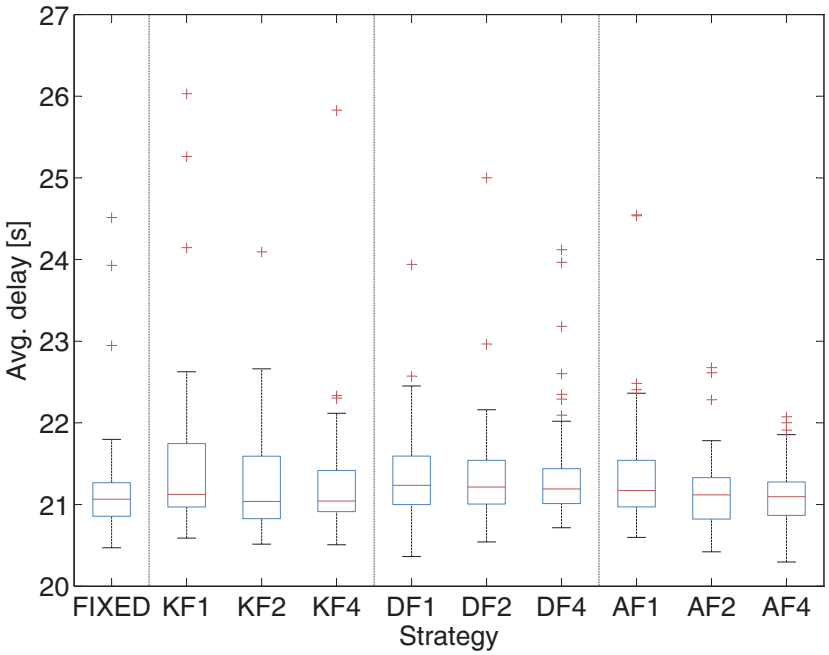


Strategy → / ↓ Percentile	C1	C2	C3	C4	C5
25th	21.20	20.98	21.04	21.12	21.13
50th	21.71	21.33	21.36	21.57	21.96
75th	22.46	21.94	21.86	24.40	24.82

Strategy → / ↓ Percentile	F1	F2	F3	F4	F5
25th	21.41	20.98	20.89	**20.86**	20.90
50th	21.87	21.56	21.12	**21.07**	21.20
75th	22.85	22.01	21.78	**21.27**	21.51

Strategy → / ↓ Percentile	I1	I2	DI1	DI2	DI3
25th	21.01	20.97	21.11	21.01	20.88
50th	21.25	21.18	21.40	21.29	21.20
75th	21.82	21.74	22.05	21.72	21.65

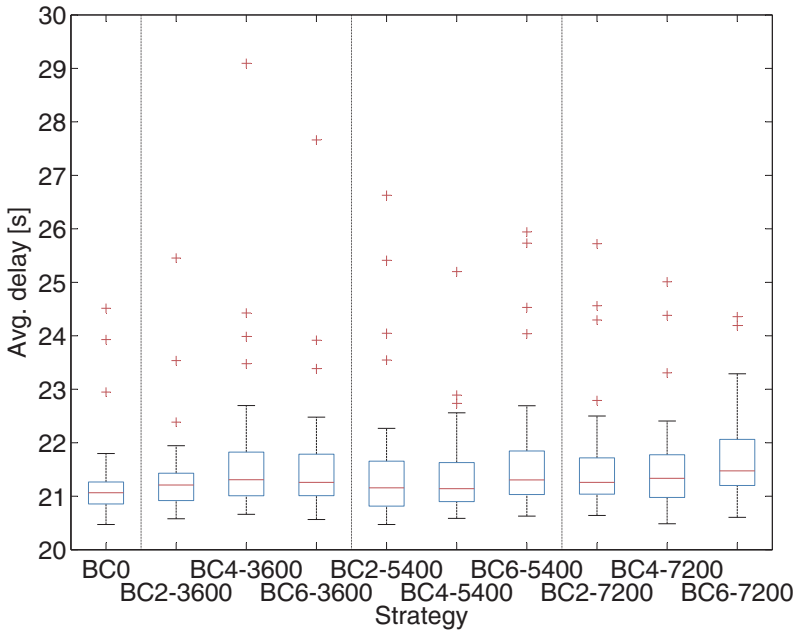
Figure 6.12: Comparison of vehicular delays for different distribution strategies



Strategy → / ↓ Percentile	FIXED	KF1	KF2	KF4
25th	20.86	20.97	20.83	20.91
50th	21.07	21.12	**21.04**	**21.04**
75th	**21.27**	21.75	21.59	21.42

Strategy → / ↓ Percentile	DF1	DF2	DF4	AF1	AF2	AF4
25th	21.00	21.01	21.01	20.97	**20.82**	20.87
50th	21.24	21.21	21.19	21.17	21.12	21.10
75th	21.59	21.54	21.44	21.54	21.33	21.28

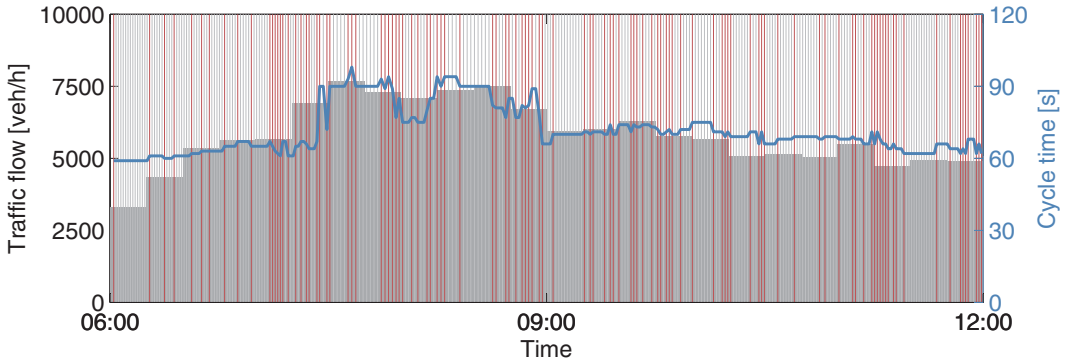
Figure 6.13: Comparison of vehicular delays for different seed handling strategies



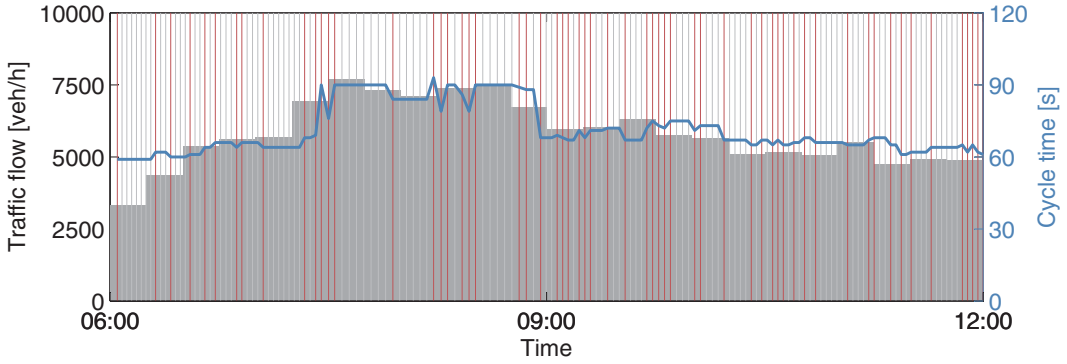
Strategy → / ↓ Percentile	BC0	BC2-	BC4- 3600	BC6-
25th	20.86	20.92	21.01	21.01
50th	**21.07**	21.21	21.31	21.26
75th	**21.27**	21.43	21.82	21.79

Strategy → / ↓ Percentile	BC2-	BC4- 5400	BC6-	BC2-	BC4- 7200	BC6-
25th	**20.82**	20.90	21.03	21.04	20.98	21.20
50th	21.16	21.14	21.30	21.26	21.34	21.47
75th	21.66	21.63	21.85	21.72	21.78	22.06

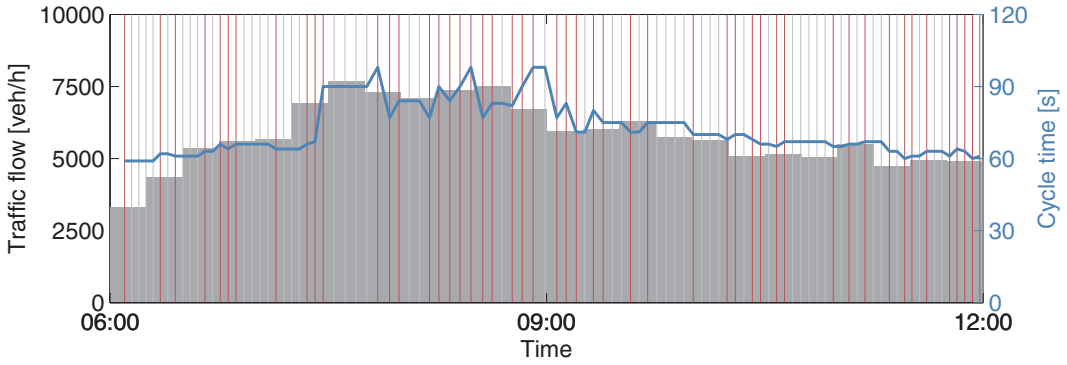
Figure 6.14: Comparison of vehicular delays for different reevaluation strategies



(a) $n_c = 1$

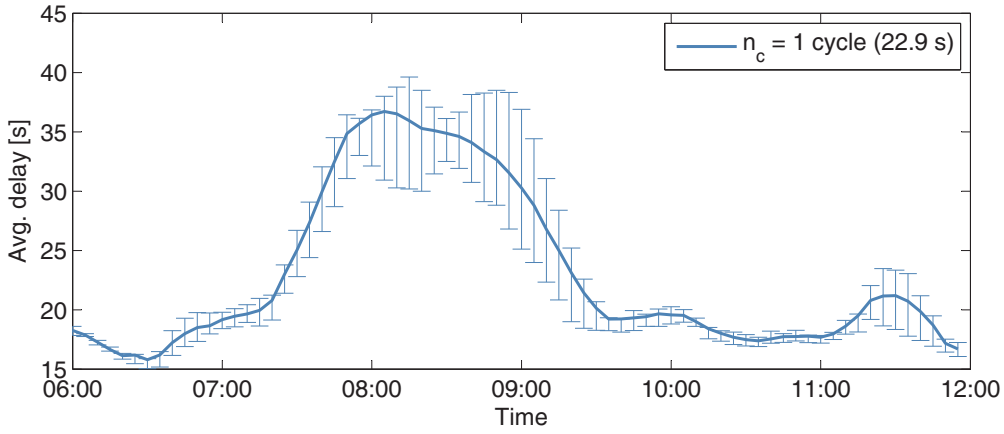


(b) $n_c = 2$

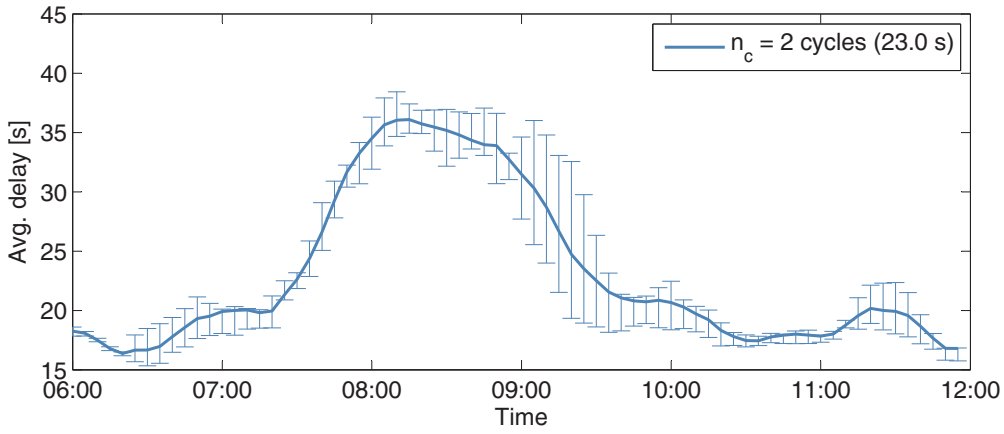


(c) $n_c = 3$

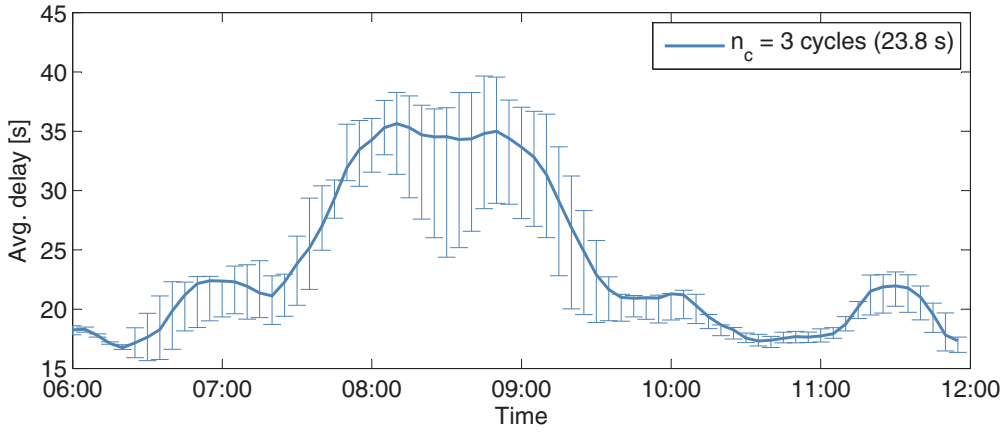
Figure 6.15: Influence of the activation interval on cycle time changes



(a) $n_c = 1$

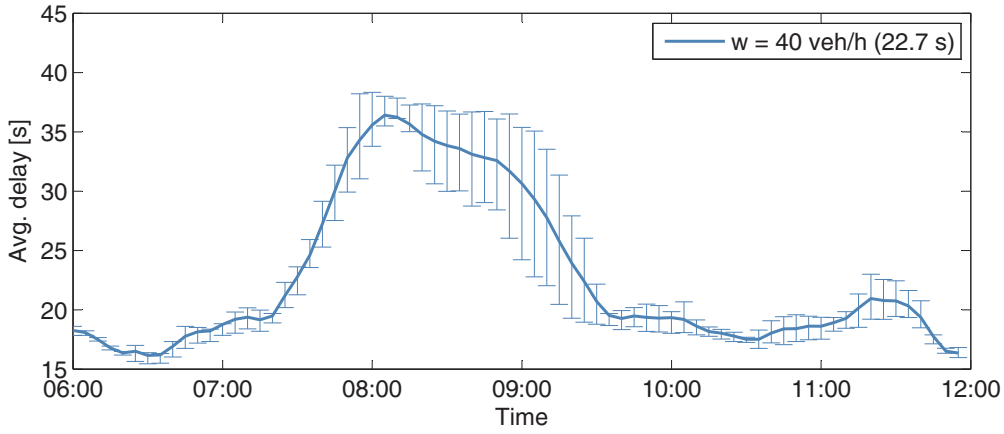


(b) $n_c = 2$

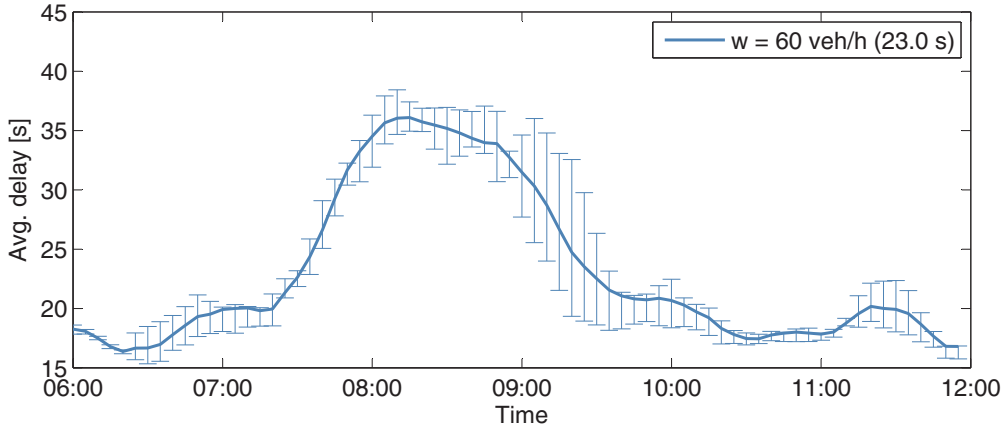


(c) $n_c = 3$

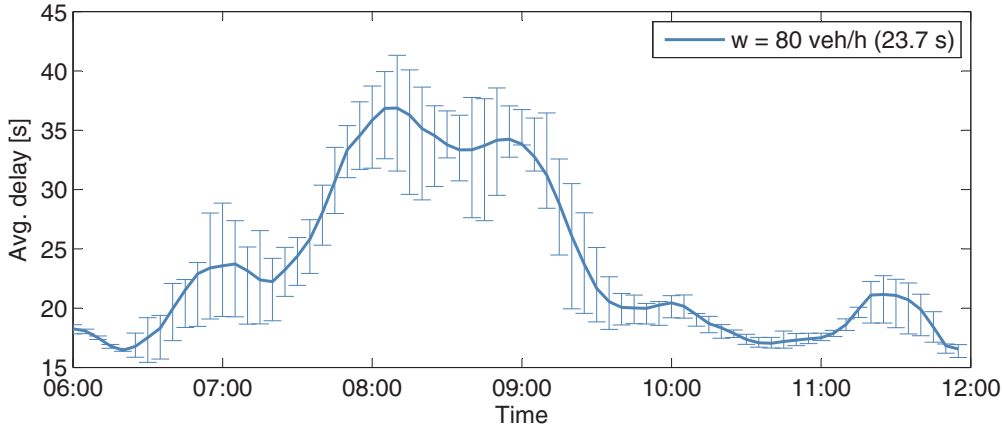
Figure 6.16: Influence of the activation interval on the vehicular delay



(a) $w = 40\,\mathrm{veh/h}$

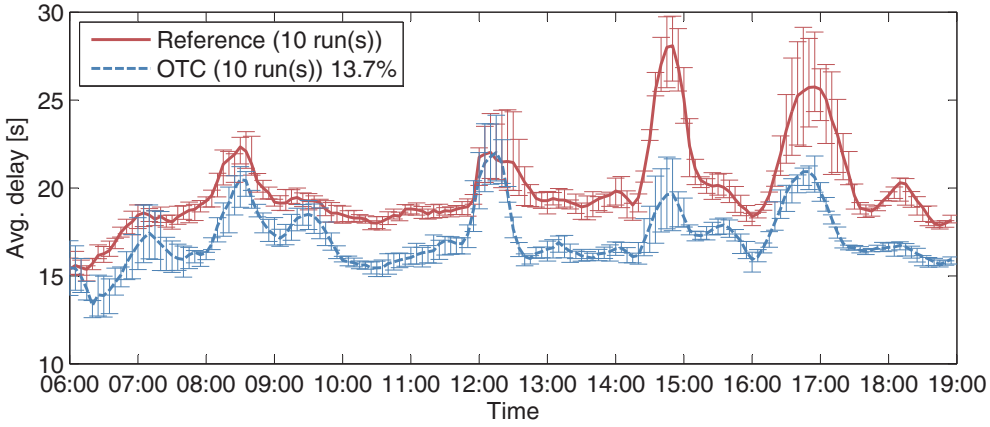


(b) $w = 60\,\mathrm{veh/h}$

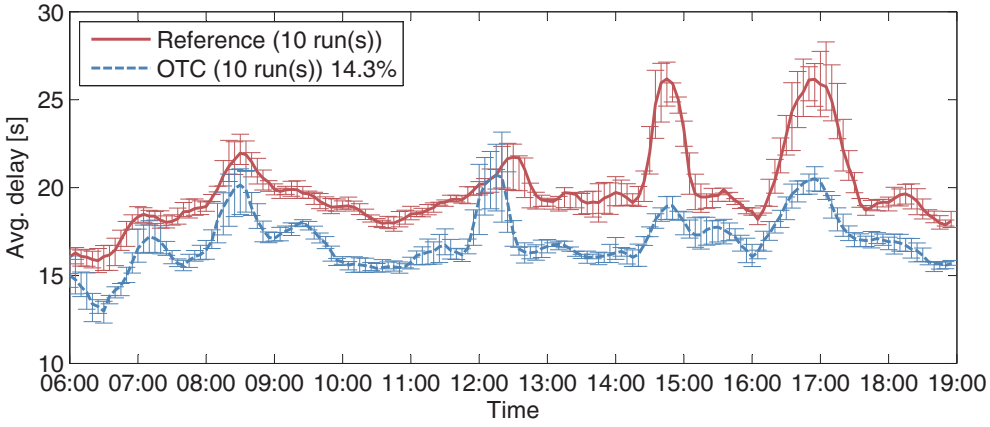


(c) $w = 80\,\mathrm{veh/h}$

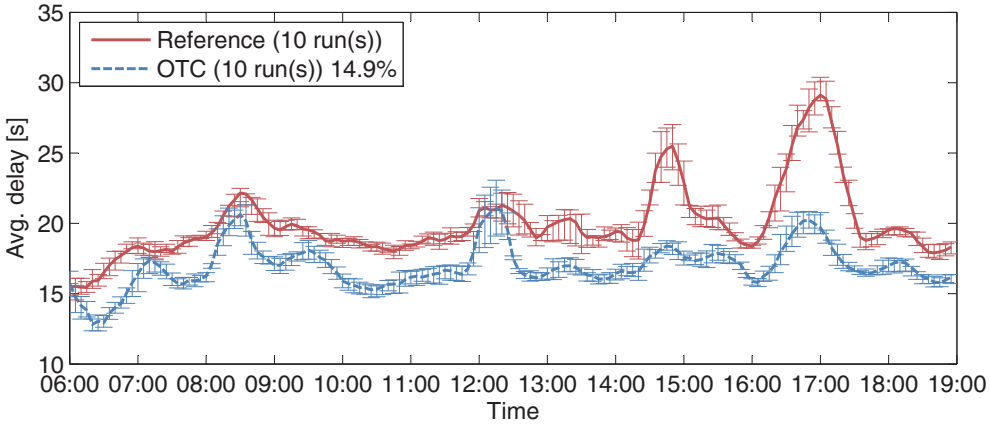
Figure 6.17: Influence of the similarity tolerance on the vehicular delay



(a) Day 1

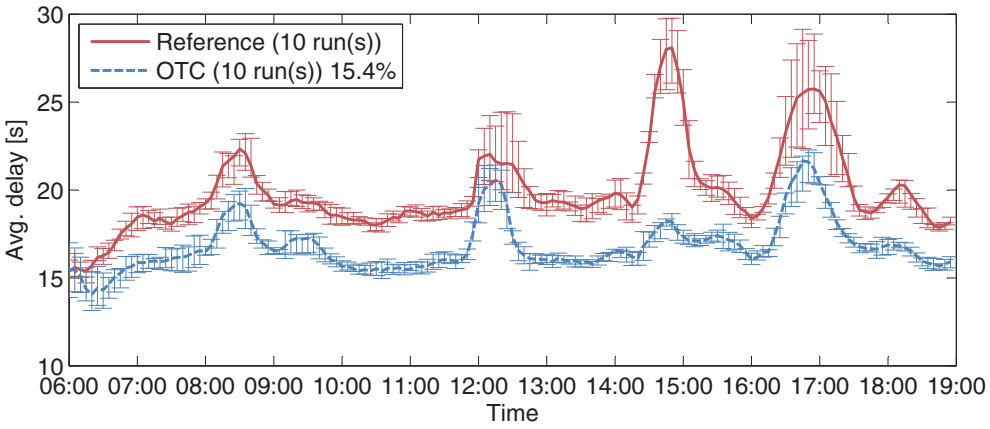


(b) Day 2

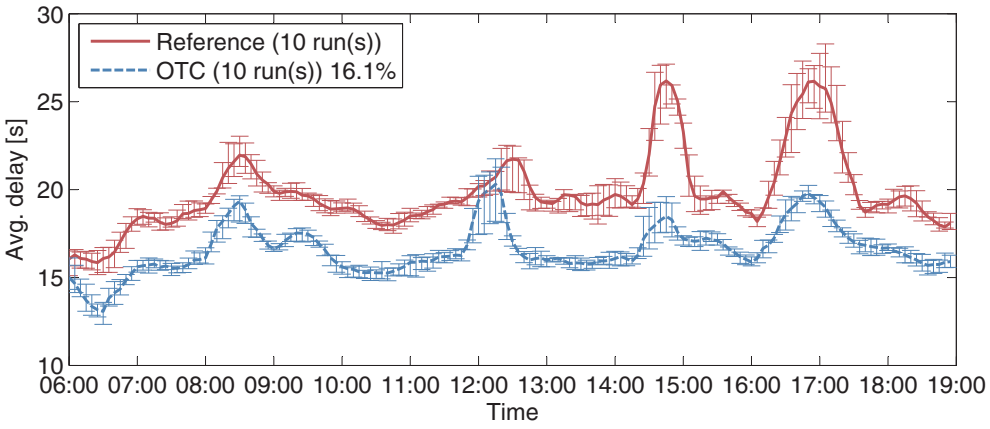


(c) Day 3

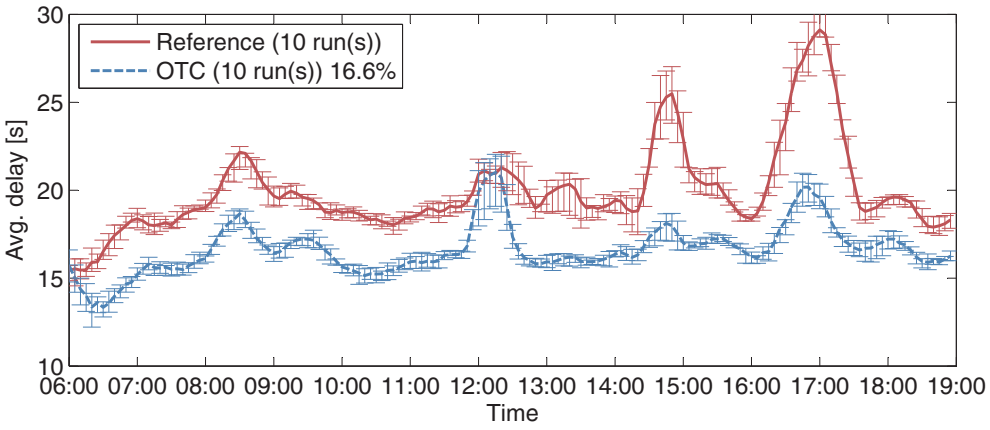
Figure 6.18: Average vehicular delay at Intersection K7 (Webster approximation)



(a) Day 1

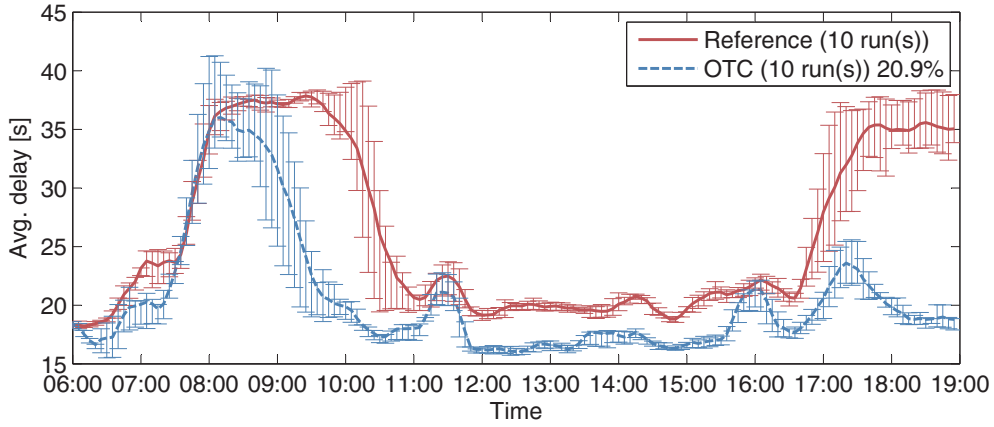


(b) Day 2

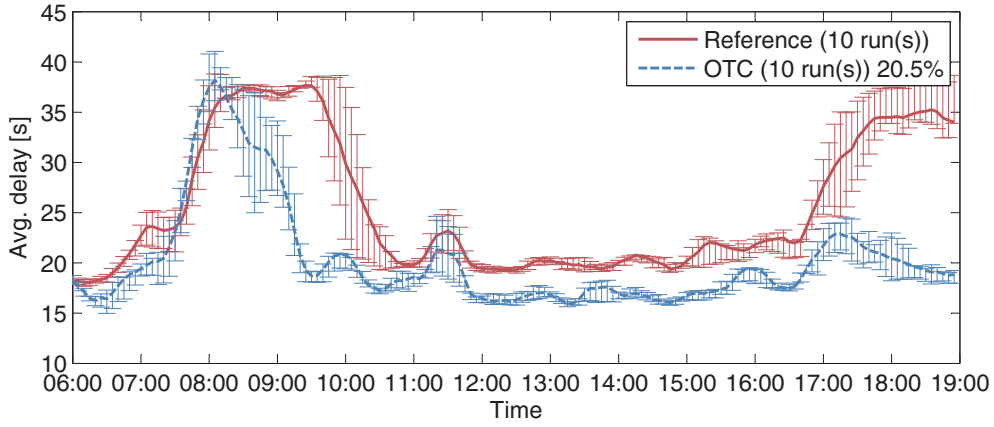


(c) Day 3

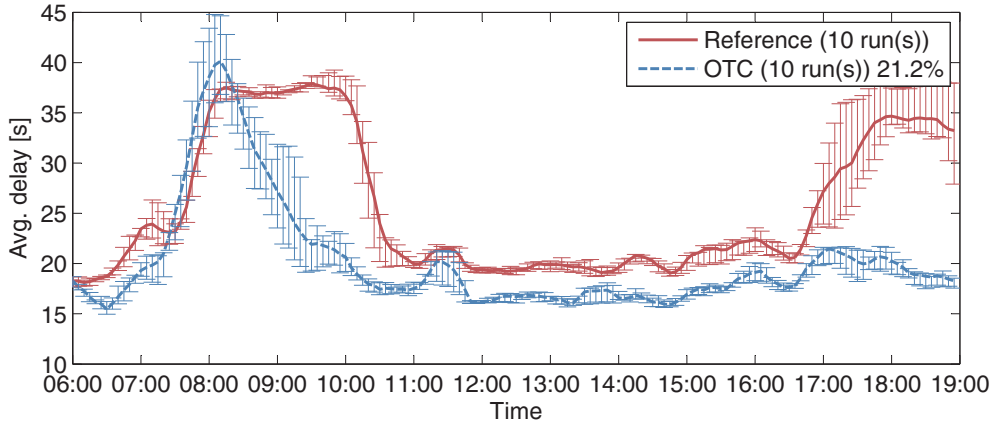
Figure 6.19: Average vehicular delay at Intersection K7 (AIMSUN simulation)



(a) Day 1

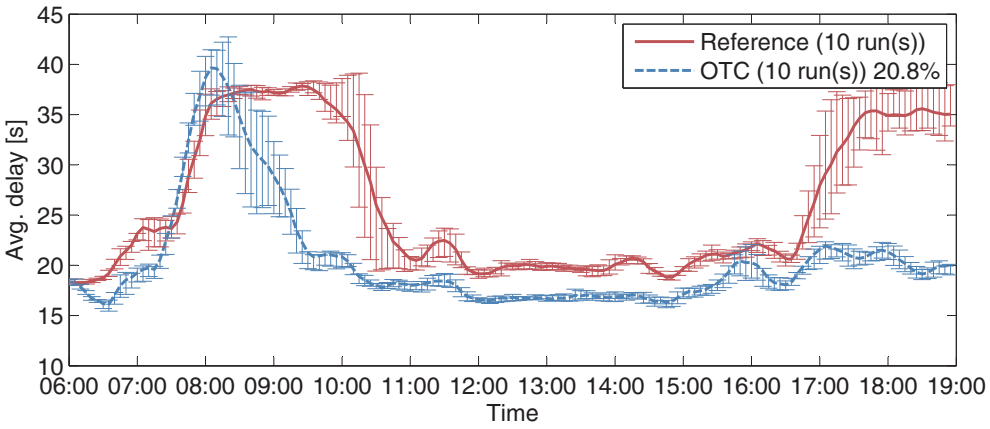


(b) Day 2

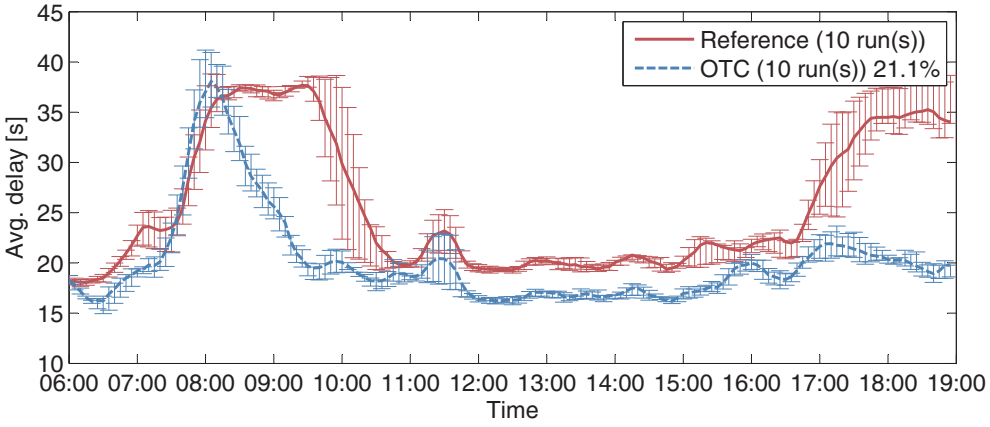


(c) Day 3

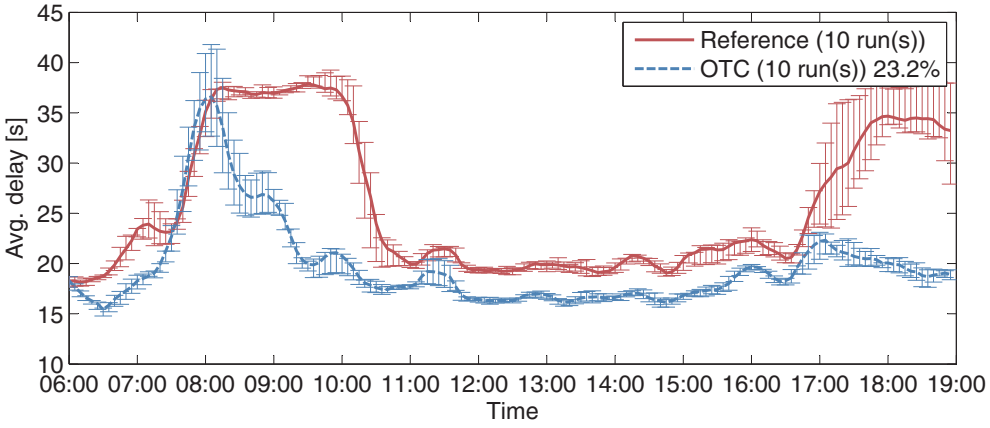
Figure 6.20: Average vehicular delay at Intersection K3 (Webster approximation)



(a) Day 1



(b) Day 2



(c) Day 3

Figure 6.21: Average vehicular delay at Intersection K3 (AIMSUN simulation)

CHAPTER 7

Decentralised coordination of organic intersections

The observer/controller evaluates and optimises an intersection's signalisation at run-time, thereby reducing the local vehicular delay. In urban road networks, where several intersections are located in close vicinity, a good signalisation does not result from appropriately configured phase durations and cycle times, only, but depends largely on another signalisation parameter: the offset.

Offsets specify the difference between a reference time and the start time of a coordinated phase (see Sections 2.2 and 2.3). By adjusting the offsets of neighbouring intersections, a *progressive signal system* (PSS, also called *green wave*, see Figure 2.3) can be established. When the offsets within a road network are optimised to suit the network's demand, a reduction of journey times, stops, and, in consequence, of fuel consumption and pollution emission can be obtained.

To decide which traffic streams in a network should be coordinated, locally available detection data is hardly sufficient. In consequence, an exchange of information among signalised intersections becomes necessary. It can be realised either implicitly (by an early detection of approaching

vehicles) or explicitly (by communication) and results in one of the following system architectures:

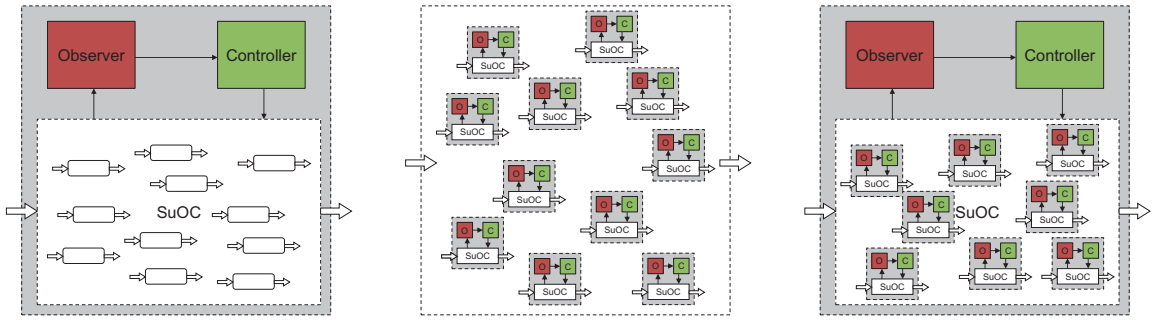
Centralised systems Many adaptive network control systems rely on a traffic control centre to adapt the network's signalisation (like, e. g., SCOOT, see Section 2.4.1). The control centre gathers detection data from the network, incorporates the data in its traffic model, computes appropriate signal timings and offsets, and adapts the signal plans accordingly. Thereby, a network-wide data pool is available to the control centre, but the necessary data communication and the centralised coordination are computationally complex, monetarily costly, and potentially susceptible to failures [74, 134]. This abets a trend towards decentralised or hierarchical systems that shift (parts of) the decision process to the intersections.

Decentralised systems A coordination of signals can also be achieved without a control centre. Decentralised approaches are implemented, e. g., by OPAC (see Section 2.4.3) and by the self-organising network control systems discussed in Section 3.3. A coordination is achieved with the help of a communication link among neighbouring intersections or by an early detection of arriving vehicle platoons. In both cases, the signalised intersections determine their signalisation based on the limited information that is locally available (which results in an implicit coordination without fixed offsets in the non-communicating case).

Hierarchical systems Hierarchical architectures are a compromise between centralised and completely decentralised systems. Some decisions are taken locally (e. g., by an intersection that adapts its phase durations), while others are taken by higher level instances (e. g., an offset optimisation conducted by a regional control centre). BALANCE and MOTION are two examples of adaptive network control systems that rely on a hierarchical system architecture (see Section 2.4.4).

Centralised, decentralised, and hierarchical system variants have also been proposed for the observer/controller [23, 190]. A single central

observer/controller (like in Figure 7.1a) is well-suited for isolated systems with a clearly defined purpose (like a signalised intersection) where the composing subsystems exhibit restricted situation and configuration spaces. Larger and more complex systems (like traffic networks) are characterised by a drastic increase of the situation and configuration space. Such systems are difficult to handle by a single centralised observer/controller due to the computational complexity of the control task. Here, a decentralised or hierarchical problem decomposition (as shown in Figures 7.1b and 7.1c) is recommendable.



(a) Central (b) Decentral (c) Hierarchical

Figure 7.1: Distribution schemes for the observer/controller architecture (based on [23])

The following chapters focus on the decentralised and hierarchical coordination of traffic signals in road networks. In the remainder of this chapter, the observer/controller architecture for traffic control is extended by a self-organising coordination mechanism. The mechanism is called DPSS (for *Decentralised Progressive Signal Systems*), relies on local communication among neighbouring intersections, and is introduced in Section 7.1. Section 7.2 summarises the results of a sensitivity study that investigates the mechanism's configuration. An experimental evaluation is in the focus of Section 7.3. The section compares the mechanism's performance to uncoordinated organic intersections. Finally, Section 7.4 concludes the discussion, before the *Regional Manager*, a hierarchical extension to the DPSS mechanism, is introduced in Chapter 8.

7.1 The DPSS mechanism

As the observer/controller framework evaluates and optimises signal plans locally at each intersection, it stands to reason that the coordination of organic intersections should also be a decentralised process. Decentralised coordination schemes eliminate the large effort that is necessary to connect intersections to a control centre and to continuously communicate detection and signalisation data between control centre and intersections.

As stated by Mück [134], centralised control is actually infeasible for some intersections due to their location. Furthermore, some networks contain only a few critical intersections that would largely benefit from adaptive control, while most intersections can be handled well by fixed-time or traffic-actuated controls. In these cases, coordinating the whole network from a control centre is not cost-effective.

Besides technical and economical reasons that favour decentralised solutions, systems that do not rely on a single centralised component – and thereby avoid a single point of failure in their architecture – can benefit from an increased robustness. Friedrich acknowledges that "distributed intelligence" can reduce the fault liability of traffic control systems, while on the other hand their adaptation to local fluctuations in the traffic flow can be improved [74].

The remainder of this section presents a decentralised collaboration mechanism for signalised intersections that is called DPSS (for *Decentralised Progressive Signal Systems*). The DPSS mechanism extends organic intersections with local communication capabilities and enables them to coordinate their signalisation in response to the network's current traffic demand. It constitutes a three step process that has been originally introduced in [145, 192].

In a first step, the network's intersections determine partners that collaborate to form a PSS. The decentralised mechanism that leads to the creation of partnerships is presented in Section 7.1.1. Once the partnerships are established, the collaborating intersections negotiate a common cycle time which is a prerequisite for coordination. The corresponding consensus algorithm is discussed in Section 7.1.2. In a third step, the partners select signal plans that respect the common cycle

time, calculate offsets, and finally establish the coordination. Section 7.1.3 presents this completing step of the coordination process.

The remainder of the section investigates traffic-responsive PSS updates in Section 7.1.4 and discusses requirements, possibilities, and limitations of the DPSS mechanism in Section 7.1.5.

7.1.1 Determining collaborating intersections

The identification of traffic streams that should be served by a PSS is a crucial step when establishing a coordinated signalisation within a road network. Coordination schemes are subject to many restrictions. The coordinated signalisation of one traffic stream often impedes the coordination of several other streams. Restrictions already apply for an arterial road where a bidirectional coordination can only be achieved in special cases due to dependencies that exist among the distances between intersections, progression speeds, and offsets [94]. The selection of traffic streams for coordination gets even more complicated in traffic networks where numerous conflicts of interest exist among the network's traffic streams. Therefore, it is important to select the coordinated streams carefully and adapt the coordination to changes in demand.

To determine a sequence of intersections that can establish a PSS improving the network's traffic flows, the DPSS mechanism applies an heuristic to identify the strongest traffic streams in a network. The heuristic considers the intersections' turning flows, relies on local communication among neighbouring intersections, and assumes that intersections have synchronised clocks.

With the help of their synchronised clocks, the signalised intersections simultaneously determine which of their motorised turning movements exhibits the strongest traffic flow. Let intersection j determine the turning from upstream intersection i to downstream intersection k as its strongest turning movement. For intersection j, it should be beneficial to coordinate the longest signal phase serving the selected turning from i to k with the respective upstream intersection i, thereby creating a *coordinated phase*. To initiate the partnership, intersection j informs its desired predecessor i that it would like to be i's successor in a PSS. After all intersections informed their desired predecessor, a local matching

takes place. Each intersection j checks whether it was chosen by its downstream intersection k as k's desired predecessor. If this is the case, j acknowledges the partnership with k. Other intersections that registered with j receive a reject message and no partnership is established with these intersections.

Based on the acknowledged partnerships, each intersection can determine whether it is part of a PSS and which of its neighbours is its predecessor or successor in the system. The first (last) intersection of a PSS knows its special position since it has no predecessor (no successor) but a successor (a predecessor). Intersections that were not integrated in a PSS did not send or receive any acknowledgements. These intersections can repeat the above process with their second most heavily used turning movement and other intersections not participating in a PSS. For all established PSSs, the collaborating intersections by now know their partners and can start to negotiate a common cycle time.

7.1.2 Determining a common cycle time

As discussed in Section 2.2.1, a common cycle time is a prerequisite to ensure that the signalisation at neighbouring intersections remains coordinated over time. Due to its influence on the coordinated intersections' capacities, the common cycle time needs to be selected carefully: For a short cycle length, the constant lost times make up a large fraction of the cycle which results in a reduced capacity of the intersection. An overly long cycle length, on the other hand, can lead to unused green times that increase the vehicular delay. Therefore, the common cycle time for the PSS has to be long enough to provide all participating intersections with sufficient capacities, while it should be as short as possible to reduce the induced delay.

Figure 2.6b exemplifies the cycle time's influence on the vehicular delay at a signalised intersection. The figure illustrates that a deviation from an optimal cycle length results in an increased delay. However, the observable increase is not symmetric. While longer cycles cause slowly increasing delays, a decrease towards the minimal cycle length results in a steep rise. Therefore, the cycle time for a PSS should equal the largest optimal cycle at a coordinated intersection.

To determine a common cycle time for the PSS that fits this requirement, each intersection i keeps tracks of its own *desired cycle time* (DCT_i) and an *agreed cycle time* (ACT) for the PSS. The desired cycle time DCT_i is the cycle length intersection i would prefer for the current traffic demand if it was not participating in a PSS. It is determined by activating the intersection's LCS for the current demand (see Section 5.4) and storing the selected signal plan's cycle length as DCT_i. As the objective of the LCS is to minimise the local vehicular delay, the determined cycle length will be close to optimal. Small deviations that result from the selection procedure are not considered critical as they lead to marginally increased delays, only (see Figure 2.6b).

The agreed cycle time ACT is the common cycle length for the coordinated intersections. It is the maximum of the desired cycle times of all intersections i in a PSS (i. e., $ACT := \max(DCT_i)$) and can be determined by a decentralised *echo algorithm* [46], assuming that each intersection i stores its knowledge on the agreed time locally as ACT_i: The first intersection in the PSS updates its desired cycle time DCT_1, sets $ACT_1 := DCT_1$, and sends ACT_1 to its successor in the PSS. The succeeding intersections i, $i = 2, \ldots, n$, where n is the last intersection in the PSS, successively update their desired cycle time DCT_i by activating their LCS, setting

$$\begin{aligned} ACT_i &:= \max(DCT_i, ACT_{i-1}) \\ &= \max\left(DCT_i, \max_{j \in \{1,\ldots,i-1\}} (DCT_j)\right) \\ &= \max_{j \in \{1,\ldots,i\}} (DCT_j), \end{aligned}$$

and sending ACT_i to the next intersection in the PSS. This process continues until the last intersection n of the PSS is reached. By then, ACT_n equals the maximum desired cycle time in the PSS. ACT_n is now propagated back to the beginning of the PSS, such that each intersection i in the PSS can replace its knowledge on the agreed cycle time by ACT_n (i. e., $ACT_i := ACT_n$ for $i = 1, \ldots, n-1$). At the end of this process, all intersections in the PSS have agreed on the same ACT that guarantees a sufficient capacity while being as short as possible.

7.1.3 Determining offsets and establishing coordination

After the intersections that participate in a PSS have been determined and all participants have agreed on a common cycle time, appropriate signal plans respecting the ACT can be selected, offsets can be calculated, and a coordinated signalisation can be established.

Selecting signal plans under cycle time constraints

For a coordinated operation, an organic intersection needs to handle the cycle time constraint defined by the agreed cycle time. Therefore, the selection mechanism of XCS-T has been modified. The modification works as discussed in Section 5.4.1, but adapts the selected signal plan's cycle by proportionally extending the plan's phase durations while keeping its phase transitions unchanged. Thereby, a signal plan with a cycle length of ACT and unchanged phase splits (apart from rounding errors) is obtained. Since the agreed cycle time is the maximum of the desired cycle times at the individual intersections within a PSS, the adapted signal plan has at least the same capacity as the original plan and is thus applicable for the observed traffic demand.

Before the adapted signal plan is activated at the intersection, it is associated with the classifiers in the action set. The classifiers are copied and the adapted plan replaces the action of each copy. The prediction error and the fitness of the copies are reset to ε_I and F_I, respectively, and all bookkeeping parameters are reinitialised as well. The only value that is kept from the original classifiers is their respective prediction. After the copies have been updated, they are included in the XCS-T population for future use. Furthermore, they replace the original action set such that their prediction values are updated during the next activation of XCS-T.

The extended classifier selection of XCS-T ensures that returned signal plans suit the current demand (by keeping the phase splits of an appropriately selected plan) while the predefined cycle time constraint is satisfied. Once the intersections of a PSS have independently chosen their signal plans, their coordination is completed by calculating appropriate offsets.

Calculating offsets

When calculating offsets for the coordinated signals, no restrictions exist for the first intersection in a PSS. For each succeeding intersection i, $i = 2, \ldots, n$, the offset o_i depends on

- the predecessor's offset o_{i-1},
- the time $d_{i-1,i}$ vehicles need to travel from intersection $i-1$ to intersection i, and
- the time q_i needed to serve queued vehicles for the coordinated phase.

Furthermore, the absolute point in time s_{abs} when the first intersection initially activates its coordinated phase must be known to all successors. Again, all necessary information is successively propagated through the PSS from intersection to intersection: The first intersection communicates s_{abs} and its offset (without loss of generality $o_1 := 0$) to its successor. In the following, the intersections i, $i = 2, \ldots, n$, successively calculate their own offset relative to the first intersection in the PSS using the formula

$$o_i := (o_{i-1} + d_{i-1,i} - q_i) \mod ACT.$$

Here, it is assumed that the time $d_{i-1,i}$ is stored locally at each intersection for all its neighbours j (one of which is intersection $i-1$). This assumption is reasonable, since $d_{i-1,i}$ can be estimated from the distance of the neighbouring intersections and the speed limits of their connecting links. The time q_i required to serve queued vehicles is derived from the average queue length.

Once the offset calculation at intersection i is finished, the values of s_{abs} and o_i are forwarded to the succeeding intersection in the PSS until the last intersection is reached and the coordination can be established.

Establishing a coordinated signalisation

To establish the calculated offset at an intersection without inappropriately interfering with the active signalisation, a transitional signal plan is activated for exactly one cycle after the currently active plan's cycle has

ended. The transitional plan is obtained by proportionally adapting the phase durations of the active plan (but keeping the phase transitions). The transitional plan's cycle time t_C is given by the equation

$$t_C := (s_{abs} + o_i - p_i - e_{abs}) \mod ACT.$$

In the equation, $s_{abs} + o_i$ defines the point in time when the coordinated phase of intersection i has to start. Correspondingly, the coordinated signal plan needs to be activated at time $s_{abs} + o_i - p_i$, assuming that p_i marks the start of the coordinated phase within the cycle. This leaves t_C seconds that have to be bridged by a transitional plan after the currently active plan has ended at time e_{abs}. If the calculated transitional cycle length t_C cannot be realised because it is shorter than the minimum cycle $t_{C,min}$, it can be redefined as $t_C := t_C + ACT$. Once the transitional signal plans have been activated and replaced at all coordinated intersections, the PSS is established.

7.1.4 Updating progressive signal systems

Due to the dynamic nature of traffic, an established coordination needs to be reassessed from time to time. Instead of periodically starting a complete recalculation of the network's coordination without considering the prevailing traffic conditions, the DPSS mechanism reacts on the following events that represent relevant changes in traffic demand:

1. A coordinated intersection can reduce its vehicular delay by changing its signal plan. Its observer/controller has identified a plan that respects the agreed cycle time of the PSS, but has a predicted vehicular delay that is more than a *signal plan tolerance* of sp_{tol} seconds below the active plan's delay.

2. A coordinated intersection i increases its desired cycle time DCT_i due to changes in traffic such that $DCT_i > ACT$.

3. A coordinated intersection i with $DCT_i = ACT$ reduces its desired cycle time due to changes in traffic.

4. For a coordinated intersection, the turning exhibiting the strongest traffic flow has changed.

In Case 1, the changing intersection i switches to the newly selected plan. Considering the start p_i' of the coordinated phase in the new plan's cycle, the plan needs to start $(p_i - p_i') \mod ACT$ seconds later than the active plan to keep the intersection's offset o_i unchanged. The necessary shift is obtained by a transitional signal plan. Within the PSS, no further changes are required. The established partnerships, the agreed cycle time, and the computed offsets remain unchanged.

For Cases 2 and 3, the agreed cycle time ACT needs to be adapted. The intersection responsible for the change announces the necessity of an ACT update to the first intersection in the PSS. As a result, this intersection starts the echo algorithm to determine a new common cycle time ACT' (see Section 7.1.2). Since implementing ACT' would temporarily disrupt the coordination and small deviations from the optimal cycle length barely affect the vehicular delay (see Figure 2.6b), the update is only executed if ACT' differs from ACT by more than an *agreed cycle time tolerance* of ACT_{tol} seconds (i. e., if $|ACT' - ACT| > ACT_{tol}$). In this case, Step 3 of the DPSS mechanism is executed (see Section 7.1.3) which results in an updated PSS with the same collaborating intersections.

In Case 4, the relative importance of traffic movements has changed such that the partnerships should be reassessed by recalculating the coordination from Step 1 (see Section 7.1.1).

7.1.5 Discussion

The DPSS mechanism that has been introduced in Sections 7.1.1 to 7.1.4 allows for a decentralised coordination of organic intersections that is adapted at run-time in response to changing traffic demands. The remainder of Section 7.1 discusses technical requirements of the DPSS mechanism, before it focuses on limitations of decentralised network control systems. The section will be concluded with a brief comparison to other self-organising approaches.

Requirements

The DPSS mechanism extends the observer/controller framework for intersections. In addition to the observer/controller's hardware require-

ments (see Section 5.5), the DPSS mechanism has its own requirements regarding traffic detection, communication, and time synchronisation.

Turning-based detection Step 1 of the DPSS mechanism assumes that traffic flows are known for each turning movement of a signalised intersection. As traffic is typically detected at signal group level – but not for individual turning movements – this assumption can be critical (e. g., when a signal group serves a shared lane that is used by several turnings). In this case, the traffic flow detected for a signal group can be either attributed to the group's main turning or it can be assigned to the different turnings according to estimated turning fractions. Thereby, the DPSS mechanism becomes applicable for signal group-based detection and induces no additional detection requirements.

Synchronised clocks To implement the DPSS mechanism, all intersections need synchronised clocks. This requirement can be fulfilled with the help of time synchronisation protocols like the *Network Time Protocol* (NTP) [129].

Local communication As all three steps of the DPSS mechanism involve the transfer of data among neighbouring intersections, a local communication capability constitutes an additional requirement. Fortunately, the DPSS mechanism is not demanding with respect to bandwidth or communication latencies. As the amount of communicated data is limited and the time span between establishing or updating PSSs is orders of magnitude larger than typical latencies for communication and processing, communication links can be implemented using standard network protocols and hardware.

Limitations

The DPSS mechanism is a decentralised heuristic for the coordination of traffic signals that is based on local traffic demands and local communication among neighbouring intersections. It has no access to a network-wide traffic model and is therefore subject to some limitations.

Within the DPSS mechanism, PSSs are derived by concatenating the strongest turning movements of neighbouring intersections (see Sec-

tion 7.1.1). The resulting coordinated traffic streams are "virtual" in the sense that the actual routes of the drivers within the network are unknown, i. e., vehicles can enter or leave a coordinated stream at any intersection. In consequence, there are special cases for which an established PSS does not coincide with an actual traffic stream in the network.

An example is given in Figure 7.2 that depicts two traffic streams that pass an arterial road of two intersections. In the depicted example, the turning from west to east is the most heavily used turning movement at both intersections. In consequence, the DPSS mechanism coordinates the signalisation along the arterial in eastbound direction although only a minority of road users actually travels the arterial from west to east. Here, the DPSS mechanism is misled and coordinates a virtual traffic stream. However, Figure 7.2 shows an illustrating example that is not typical for real-world networks. In the vast majority of cases, a coordination established by the DPSS mechanism will coincide with the network's traffic streams.

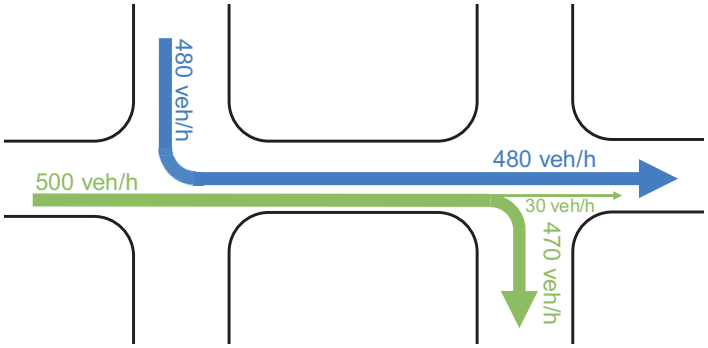


Figure 7.2: Limitations of the DPSS mechanism

Assuming that the network's traffic streams and the established coordination coincide, the greedy determination of collaborating partners leads to another limitation. Step 1 of the DPSS mechanism (discussed in Section 7.1.1) treats the strongest traffic stream within a network preferentially. As a coordinated signalisation of the strongest stream can impede the coordination of several other streams that in sum serve more vehicles, this greedy approach can lead to a suboptimal coordination with respect to the network-wide number of stops. This issue is discussed

in more detail in Chapter 8, where it is addressed by an hierarchical extension to the DPSS mechanism.

Despite the inherent limitations of a decentralised heuristic, the DPSS mechanism allows for a traffic-responsive signal coordination that reduces the network-wide number of stops. Before the achievable reduction is experimentally evaluated, the DPSS mechanism will be situated in the context of other adaptive network control systems.

Comparison

Unlike the centralised or hierarchical network control systems discussed in Section 2.4, the DPSS mechanism does not require a computationally complex network-wide offset optimisation at a control centre to establish a coordinated signalisation. All decisions regarding signal timings and offsets are taken locally at the intersections. The DPSS mechanism thereby avoids the cost and effort necessary for continuously communicating detection and signalisation data between intersections and control centre and allows for adaptive signal control in networks where centralised system architectures are infeasible or not cost-effective [134]. Furthermore, a potential fault liability is reduced by eliminating the control centre as single point of failure [74].

Due to its decentralised working principle, the DPSS mechanism is closely related to other self-organising traffic systems. It bears similarities to Bazzan's game-theoretic approach (see Section 3.3.2) that achieves coordination with the help of communicating intersections. Bazzan's intersections select their signal plan from a predefined set in response to local demands and play coordination games to establish a coordinated signalisation. In contrast to this game-theoretic approach, the DPSS mechanism learns and optimises its signal plans at run-time (and is therefore not restricted to a predefined set of plans). Furthermore, it does not require any prior knowledge on coordination options (which is a prerequisite for the definition of payoff matrices).

Other decentralised traffic control systems like OPAC (discussed in Section 2.4.3), the SOTL mechanism of Gershenson et al., or the pressure-based approach of Lämmer et al. (both discussed in Section 3.3.1) do not rely on a communication link, but achieve an implicitly coordinated

signalisation by reacting on vehicle platoons. Reactions are based on predefined switching schemes and require an early detection of arriving platoons. In contrast, the DPSS mechanism establishes an explicit coordination of neighbouring intersections that optimise their signal plans at run-time. Both system philosophies have specific advantages and drawbacks: While the platoon-based signalisation of non-communicating intersections helps to reduce delays and stops, signalisation changes are no longer predictable for road users and human planners. Furthermore, the apparent benefit of a communication-free system comes at the cost of sophisticated vehicle detection capabilities at each intersection.

Compared to the discussed state-of-the-art systems, the DPSS mechanism is novel as it combines an on-line learning and optimisation of signal plans with an explicit traffic-responsive coordination of neighbouring intersections that is achieved without a traffic control centre. In the following section, the DPSS mechanism will be configured based on the results of a sensitivity study before its performance is evaluated in simulation.

7.2 Sensitivity study

The DPSS mechanism introduces several potential design factors that affect the performance of an established coordination:

Check frequency A coordinated signalisation within a road network has to be adapted to changes in the network's traffic demand. Therefore, the DPSS mechanism periodically checks and updates established PSSs (see Section 7.1.4). The frequency of these checks affects the mechanism's reactivity in the presence of changes.

Agreed cycle time tolerance (ACT_{tol}) When established PSSs are reassessed by the DPSS mechanism, the coordinated intersections renegotiate their agreed cycle time. If the new agreed cycle time ACT' differs from the currently established cycle length ACT by more than ACT_{tol} seconds, the PSS is updated. The update requires transitional signal plans that temporarily disrupt the coordination (see Section 7.1.3). This results in a trade-off between the ben-

efit gained from the update and the loss due to the temporary disruption.

Signal plan tolerance (sp_{tol}) Another parameter that influences the reassessment of an established coordination is the signal plan tolerance. The tolerance defines a threshold for local signal plan changes (see Section 7.1.4). If a different signal plan with the same cycle length is expected to reduce the local vehicular delay by more than sp_{tol} seconds, the plan is changed. However, the change temporarily disrupts the coordination of the affected intersection such that sp_{tol} has to be selected carefully.

Before the potential design factors are investigated in more detail, the test network considered in the simulation study is presented in Section 7.2.1. The frequency of coordination checks is in the focus of Section 7.2.2, while a factor screening experiment determines the configuration of the agreed cycle time tolerance ACT_{tol} and the signal plan tolerance sp_{tol} in Section 7.2.3.

7.2.1 Test case

The test network used in the sensitivity study consists of five signalised intersections that are located in 250 m to 350 m distance along an arterial road (see Figure 7.3). The intersections support twelve turning movements. Their approaches are one-laned, but the arterial road segments provide an additional side-lane for left-turns. Each intersection is operated by a three-phased fixed-time signal plan. Traffic leaving the arterial by a left-turn is served in Phase 1, arterial traffic and all vehicles turning right to leave the arterial are handled by Phase 2. Phase 3 serves traffic arriving from the side roads. The intersections are equipped with an observer/controller that adapts their signalisation at run-time. The observer/controller is configured based on the findings in Sections 5.3, 6.4, and 6.5, its configuration is summarised in Table 7.1.

A sequence of varying traffic demands with a total duration of four hours has been implemented to provide a dynamic environment for the sensitivity study (see Table 7.2). While the most heavily used origin/destination (O/D) pair is $E \rightarrow W$ (see Figure 7.3 for O/D labels)

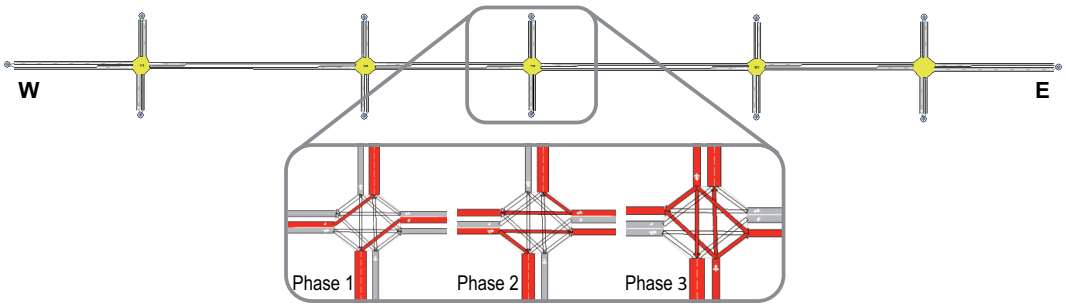


Figure 7.3: An arterial road with three-phased intersections

Table 7.1: Observer/controller configuration for the sensitivity study

Design factor	Factor level
Observer	
Detection interval	$n_e = 10$
Controller – Level 1	
Start population	Empty
Prediction, error, and fitness	$\rho_I \leftarrow$ EA estimation, $\varepsilon_I = 25$, $F_I = 0.01$
Deletion factors	$N = 200$, $\theta_{del} = 20$, $\delta = 0.1$
Reinforcement factors	$\beta = 0.2$, $\alpha = 0.1$, $\varepsilon_0 = 2$, $\nu = 5$
XCS-T factors	$n_c = 2$, $w = 60$
Controller – Level 2	
Start population	Random
Population size	$\mu = 16$, $\lambda = 24$
Selection operator	$(\mu + \lambda)$
Evaluation function	Webster approximation
Stopping criterion	1552 evaluations (i. e., 64 generations)

during the first half of the simulation, the predominant traffic direction is reversed with the beginning of the simulation's second half. After the first and third simulated hour, the traffic demand in the network is increased, but the predominant traffic direction is kept. For a coordinated operation, the changes in demand require local signal plan updates that also affect the agreed cycle time of an established PSS. Furthermore, the organic intersections need to update their partnerships after the first half of the simulation period.

Table 7.2: Traffic demands for the arterial road network

Traffic demands for O/D pairs (in veh/h)			1st hour	2nd hour	3rd hour	4th hour
W	$\rightarrow$	E	250	300	500	650
E	$\rightarrow$	W	500	650	250	300
Others			10	10	10	10
Total			2050	2250	2050	2250

For the sensitivity study, the arterial road network has been simulated in AIMSUN v. 5.1.11 running under Microsoft Windows Vista 64-bit on a 2.5 GHz Intel Core 2 Quad processor equipped with 8 GB RAM. The four available CPU cores are shared among the network's five signalised intersections.

7.2.2 Update frequency

The timely detection of changes in demand is a prerequisite for a traffic-responsive signalisation. In the DPSS mechanism, established PSSs are periodically reassessed based on the checklist in Section 7.1.4. As changes are initiated only when the relative importance of traffic streams has changed or when the tolerances ACT_{tol} or sp_{tol} are exceeded, a reassessment can be performed frequently without running the risk of unnecessarily disrupting an established coordination.

Figure 7.4 exemplifies the reassessment of a coordinated signalisation for the arterial road network which has been simulated for a check frequency of five minutes with tolerances $ACT_{tol} = 5\,\text{s}$ and $sp_{tol} = 10\,\text{s}$. A red vertical line indicates changed partnerships, green lines depict changes

of the agreed cycle time (solid line) or of a local signal plan (dashed line). Grey vertical lines represent reassessments that have not changed the coordination. Additionally, the agreed cycle time of an established coordination is shown in blue, while grey bars in the background visualise the simulated traffic volume.

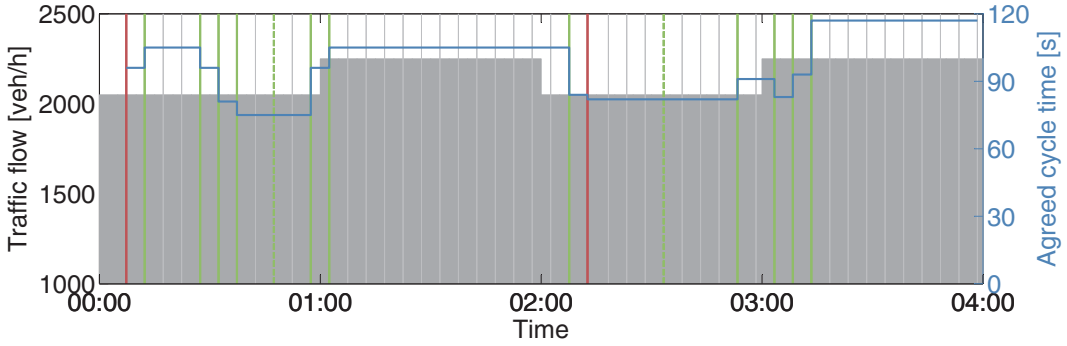


Figure 7.4: Coordination updates for the arterial road network (5 min check frequency, $ACT_{tol} = 5\,\text{s}$, and $sp_{tol} = 10\,\text{s}$)

The figure shows that coordination partnerships of the arterial's intersections change twice in the course of the simulation. The first change occurs shortly after the start of the simulation when the arterial is coordinated in westbound direction. Another change in the beginning of the simulation's second half establishes an eastbound coordination. Both changes are reasonable when considering the network's traffic demands given in Table 7.2.

Once a PSS has been established, its agreed cycle time is occasionally updated. Especially during the first simulated hour, updates are performed rather frequently because the local observer/controller components are still populating their mappings. Once these start-up effects are overcome, the agreed cycle time roughly follows the network's traffic demands. Shortly after a decrease (or increase) of demand, the agreed cycle time is decreased (or increased) to accommodate the change. In periods of constant demands, a reassessment does usually not affect the agreed cycle time. Local signal plan changes not affecting the coordination are performed rather infrequently due to the large signal plan tolerance used for the example. The influence of different tolerance levels on the performance of the DPSS mechanism is now investigated in more detail.

7.2.3 Agreed cycle time tolerance and signal plan tolerance

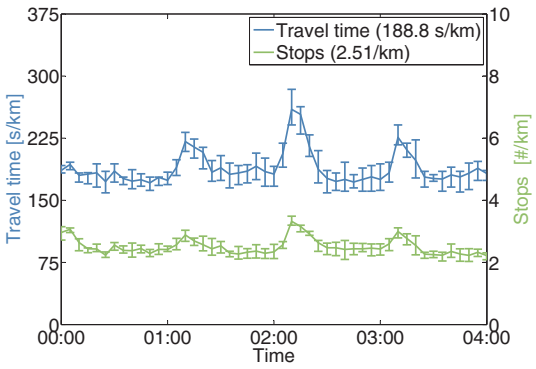
Whether or not the network's signal coordination is updated during a reassessment depends on the chosen levels for the agreed cycle time tolerance and the signal plan tolerance. Both tolerances have to be selected carefully: If their levels are too small, minor fluctuations are sufficient to trigger frequent coordination updates that are accompanied with disruptions caused by transitional signal plans. Overly large tolerances, on the other hand, inhibit a sensitive reaction on changing demands. The trade-off is investigated in a sensitivity study.

The study is conducted for the arterial road network and considers the agreed cycle time tolerance ACT_{tol} and the signal plan tolerance sp_{tol} as design factors. Both factors are investigated in a factorial screening experiment for the factor levels $ACT_{tol} \in \{2\,\mathrm{s}, 5\,\mathrm{s}, 10\,\mathrm{s}\}$ and $sp_{tol} \in \{5\,\mathrm{s}, 10\,\mathrm{s}\}$. The network-wide travel time and the network-wide number of stops serve as response variables that have to be minimised. Both response variables have been averaged over ten simulated replications with different random seeds to attribute for the stochasticity of the experimental setup.

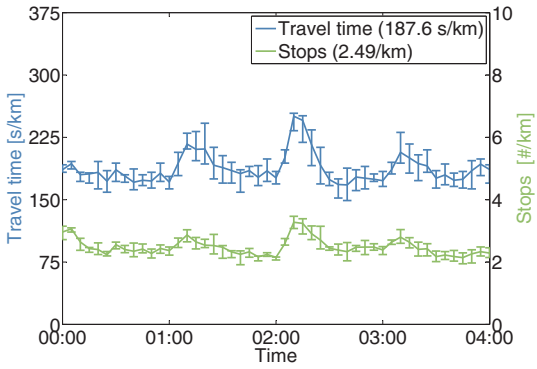
Figure 7.5 summarises the results of the sensitivity study. The figure depicts the travel times and stops recorded over the simulation period. Results for each factor combination are averaged over the ten conducted replications, additional error bars indicate the measurements' 25th and 75th percentiles.

The development of travel times and stops over the course of the simulation follows a pattern that is similar for all factor combinations. In the beginning, both response variables can be reduced as the observer/controller components populate their initially empty mappings and establish a coordinated signalisation. Temporarily increased travel times and stops are observed after the hourly changes in the simulated traffic demand as the organic intersections need to adapt their signalisation. The increase is most apparent in the beginning of the simulation's second half as here the predominant traffic stream is abruptly changed such that new coordination partnerships are required.

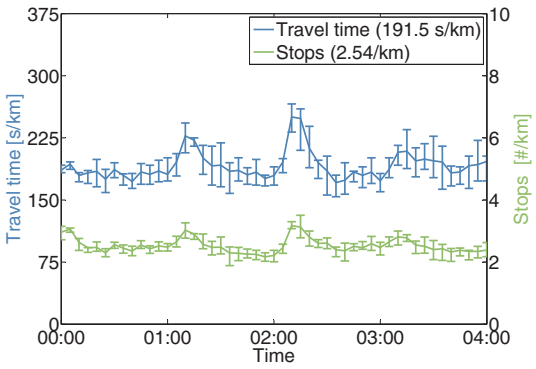
Considering the obtained travel times and stops, conclusions regarding recommended tolerance levels can be drawn. For local signal plan



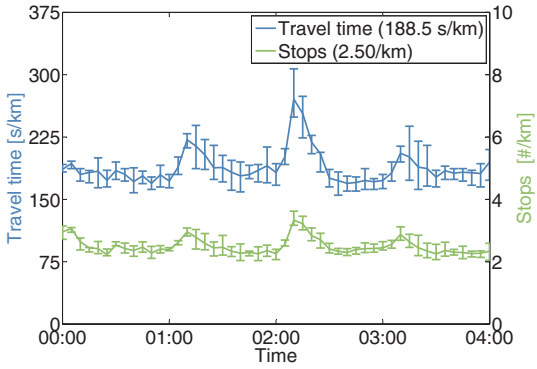
(a) $ACT_{tol} = 2\,\mathrm{s}$, $sp_{tol} = 5\,\mathrm{s}$



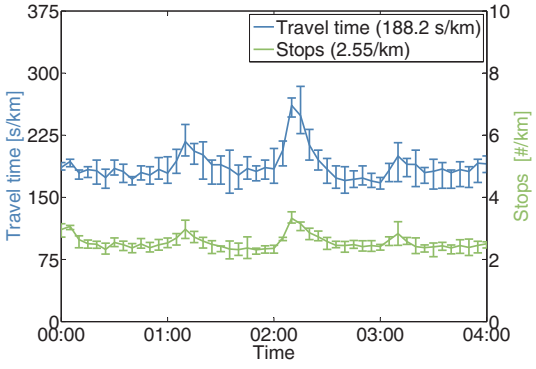
(b) $ACT_{tol} = 2\,\mathrm{s}$, $sp_{tol} = 10\,\mathrm{s}$



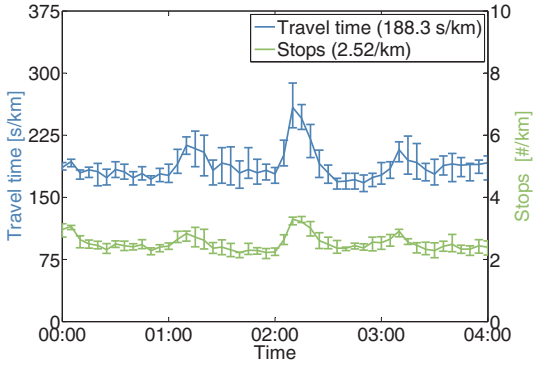
(c) $ACT_{tol} = 5\,\mathrm{s}$, $sp_{tol} = 5\,\mathrm{s}$



(d) $ACT_{tol} = 5\,\mathrm{s}$, $sp_{tol} = 10\,\mathrm{s}$



(e) $ACT_{tol} = 10\,\mathrm{s}$, $sp_{tol} = 5\,\mathrm{s}$



(f) $ACT_{tol} = 10\,\mathrm{s}$, $sp_{tol} = 10\,\mathrm{s}$

Figure 7.5: Travel times and stops for the arterial road network

changes, a relatively large tolerance of $sp_{tol} = 10\,\text{s}$ is beneficial. When comparing factor combinations with the same agreed cycle time tolerance but different signal plan tolerances, lower travel times and fewer stops are obtained for (nearly) all experiments that apply $sp_{tol} = 10\,\text{s}$ instead of $sp_{tol} = 5\,\text{s}$. The only exception are the two factor combinations depicted in Figures 7.5e and 7.5f, where the travel time is slightly increased by 0.1 s/km.

Small signal plan tolerances decrease the threshold for local signal plan changes. Local changes affect the signalisation of a single coordinated intersection, but do not have any effect on the agreed cycle time or the partnerships within a PSS. Nevertheless, the study shows that the threshold for local changes should be large, as XCS-T's signal plan selection is based on relatively inaccurate delay predictions when signal plans have to be adapted to suit a given cycle time constraint (see Section 5.4.1 for a detailed discussion of XCS-T's signal plan selection under cycle time constraints).

Regarding the agreed cycle time, Figure 7.5 shows that an increasing tolerance is accompanied with an increasing number of stops within the network. The observation holds independent of the chosen signal plan tolerance (compare Figures 7.5a, 7.5c, and 7.5e and Figures 7.5b, 7.5d, and 7.5f, respectively). The observed travel times do not exhibit a clear relation to the cycle time tolerance. As the lowest travel times are obtained for the same factor combination that also exhibits the lowest number of stops (i. e., $ACT_{tol} = 2\,\text{s}$ and $sp_{tol} = 10\,\text{s}$, see Figure 7.5b), this combination is used in the experimental evaluation of the DPSS mechanism.

7.3 Comparison to uncoordinated organic intersections

To assess the DPSS mechanism, the performance of coordinated and uncoordinated organic intersections has been compared in a simulation study. Section 7.3.1 describes the investigated test case and the experimental setup, before evaluation results are discussed in Section 7.3.2.

7.3.1 Test case

The comparison of coordinated and uncoordinated organic intersections has been conducted for the Manhattan network depicted in Figure 7.6. The network consists of six intersections that each support twelve turning movements. The connecting road segments have a length of 250 m to 350 m, are two-laned, and provide an additional side-lane for left-turns. Each intersection is controlled by an observer/controller that reconfigures a fixed-time signal plan. The plan combines the intersection's eight signal groups in four phases. Phases 1 and 3 serve left-turning vehicles, while Phases 2 and 4 serve vehicles going straight ahead or turning right (see Figure 7.6).

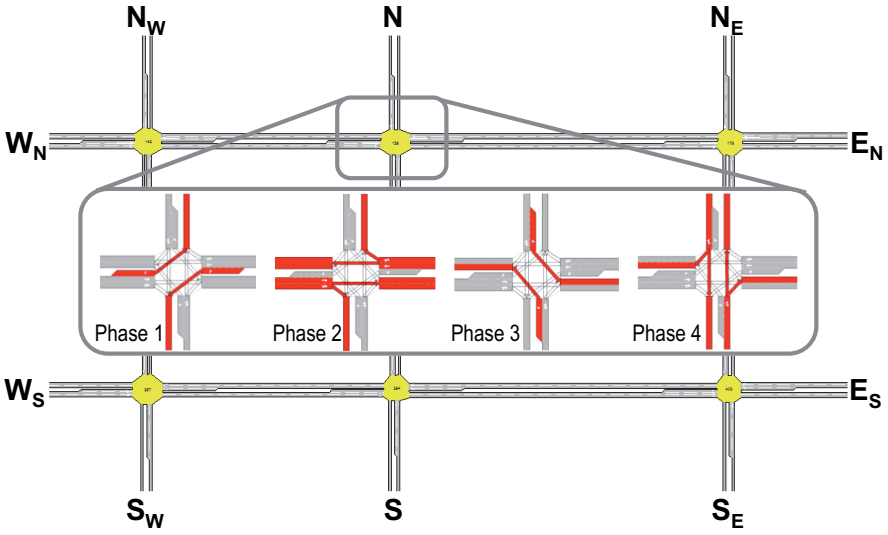


Figure 7.6: A Manhattan network with four-phased intersections

In the simulation study, the network has been simulated for four hours with varying demands. Throughout the complete simulation period, several strong traffic streams travel the network in east-, west-, north-, and southbound directions. Changes in their demand have been designed to resemble the real-world where peak hours and changing major travel directions (to and from the city centre) can be observed.

Table 7.3 summarises the simulated demand for the Manhattan network. During the first simulated hour, the eastbound streams (starting at origins W_N and W_S and ending at destinations E_N and E_S, respectively)

are most heavily travelled. This remains unchanged for the second hour during which the initial demand is increased by 25 %. With the beginning of the third simulated hour, the traffic demand is reset to its initial level, but the most heavily travelled streams change their directions. Finally, the demand is again increased for the fourth hour of a simulation.

Table 7.3: Traffic demands for the Manhattan network

Traffic demands for O/D pairs (in veh/h)			1st hour	2nd hour	3rd hour	4th hour
W_N	$\rightarrow$	E_N	900	1125.0	450	562.5
E_N	$\rightarrow$	W_N	450	562.5	900	1125.0
W_S	$\rightarrow$	E_S	900	1125.0	450	562.5
E_S	$\rightarrow$	W_S	450	562.5	900	1125.0
N_W	$\leftrightarrow$	S_W	350	437.5	350	437.5
N	$\leftrightarrow$	S	350	437.5	350	437.5
N_E	$\leftrightarrow$	S_E	350	437.5	350	437.5
Others			10	12.5	10	12.5
Total			5600	7000	5600	7000

The Manhattan network provides the test case for a simulation-based comparison of coordinated and uncoordinated organic intersections. For the comparison, the intersections' observer/controller components have been configured based on the findings of the previous sensitivity studies. Table 7.1 summarises the applied configuration. For coordinated intersections, the DPSS mechanism uses a check frequency of five minutes, an agreed cycle time tolerance of $ACT_{tol} = 2\,\text{s}$, and a signal plan tolerance of $sp_{tol} = 10\,\text{s}$. This configuration corresponds to the recommendations in Section 7.2.

Comparisons are based on the average travel time and the average number of stops observed for the network and for selected traffic streams. Additionally, fuel consumption and pollution emissions have been evaluated. The fuel consumption of simulated vehicles is calculated with the help of AIMSUN's *fuel consumption model* [196]. Calculations are based on consumption rates at constant speeds of 90 km/h and 120 km/h and incorporate data from an additional factor table to account for the acceleration, deceleration, and idling of simulated vehicles. Consumption rates

and factor tables have been configured based on data taken from [197] (consumption rates) and [67] (factor tables). For this setup, the model reflects the fuel consumption of vehicles built in 1994. Unfortunately, more recent data has not been available. Since 1996, the fuel consumption of vehicles sold in the EU is determined according to the *New European Driving Cycle* (NEDC, EU Directives 80/1268/EWG and 93/116/EWG) that no longer incorporates the required consumption rates at constant speeds.

As a vehicle's Carbon Dioxide (CO_2) emission is (for a given type of fuel) directly proportional to its fuel consumption [198], the emission of this harmful greenhouse gas can be estimated using data provided by the fuel consumption model. Other pollutants like Carbon Monoxide (CO), Nitrogen Oxides (NO_x), and un-burnt Hydrocarbons (HC) have to be treated by a separate *pollution emission model*, as their emission is not directly linked to fuel consumption. CO, NO_x, and HC are emitted especially during high load and idling periods of petrol and diesel engines (i. e., when vehicles are standing with running engines or when they have to accelerate after a stop). All three pollutants are harmful to the human health. CO reduces the blood's Oxygen carrying capacity, while HC and NO_x affect the human respiratory system and contribute to ground level Ozone formation [198]. Therefore, the emission of the three pollutants has been estimated with the help of AIMSUN's pollution emission model [196] using data available in [149].

In the following, comparative results based on the above mentioned response variables are presented. As in the preceding sensitivity study (Section 7.2), all simulations have been conducted using AIMSUN v. 5.1.11 running under Microsoft Windows Vista 64-bit on a 2.5 GHz Intel Core 2 Quad processor equipped with 8 GB RAM. The four available CPU cores are shared among the network's signalised intersections.

7.3.2 Simulation results

The test network has been simulated for coordinated and uncoordinated organic intersections. In the uncoordinated case, the observer/controller components adapt the signal plans of their respective intersections independently. In the coordinated case, intersections apply the DPSS mechanism and collaborate to set up PSSs.

For the Manhattan network, the DPSS mechanism establishes two parallel PSSs. In the first half of the simulation, the two eastbound traffic streams are most heavily travelled and are thus served by PSSs. When the traffic demand changes at the beginning of the simulation's second half, the DPSS mechanism adapts the coordination accordingly and establishes two PSSs for the westbound streams.

Network-wide travel times and stops

Figure 7.7 illustrates how the coordination affects the network by comparing network-wide travel times and stops for the coordinated and the uncoordinated case. Depicted results are averaged over ten simulation runs with different random seeds to attribute for stochastic influences in the experimental setup. Error bars are omitted for improved readability.

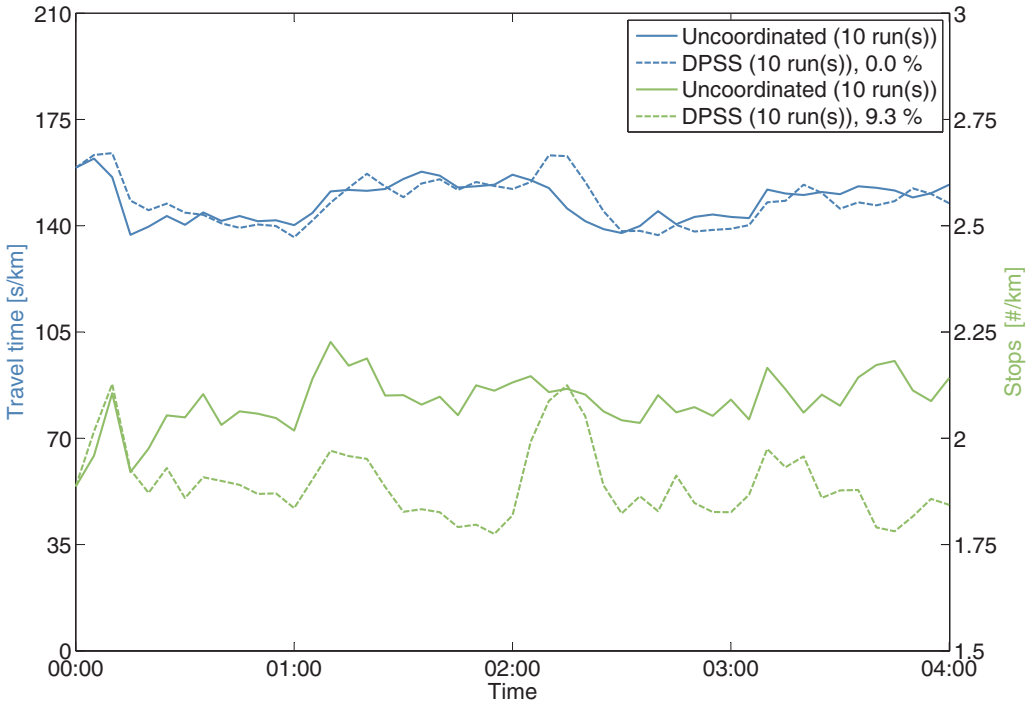


Figure 7.7: Travel time and stops for uncoordinated and coordinated signalisation (DPSS) in a Manhattan network

Compared to an uncoordinated signalisation, the DPSS mechanism reduces the network-wide number of stops for nearly the whole simulation

period. Exceptions occur in the beginning of the simulation's first and second half when the number of stops reaches a level similar to that of uncoordinated intersections. During these periods, a coordinated signalisation has not yet been established (first half) or is not yet adapted to changed demands (second half). Despite the temporary decline, the coordination reduces the network-wide stops by 9.3 % over the simulation period.

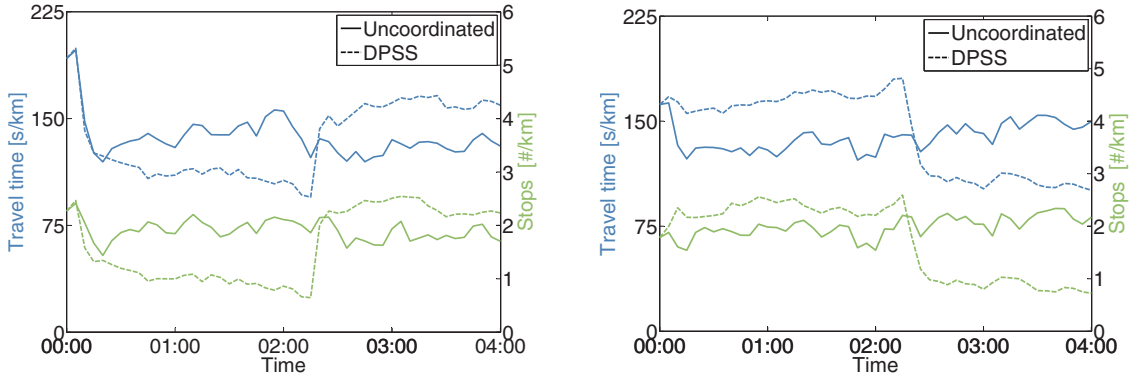
Regardless of this considerable reduction, the network-wide travel times are mostly unaffected by coordination. Figure 7.7 shows that travel times obtained for the DPSS mechanism fluctuate at the level of an uncoordinated signalisation for most of the simulation period. An increase can be observed for the beginning of the simulation's second half, when the coordinated signalisation has to be adapted to changed demands, but for the complete simulation period travel times are neither increased nor reduced.

The DPSS mechanism's effect on travel times is limited as the coordination tends to increase the coordinated intersections' cycle length (see Section 7.1.2). Instead of being operated with their optimal cycle, coordinated intersections have to utilise the longer agreed cycle time of their PSS. This increases their vehicular delay and limits the observer/controller's flexibility to react on local fluctuations in traffic. Delay savings due to a reduced number of stops for the coordinated phase can compensate this increase, but cannot reduce the vehicular delay below the uncoordinated level. In consequence, travel times cannot be reduced by the same amount that is achieved for stops.

Travel times and stops for selected traffic streams

While Figure 7.7 visualises travel times and stops for the complete network, Figure 7.8 illustrates the effect of a coordinated signalisation on traffic streams. The figure depicts travel times and stops for both directions of the network's northern arterial. While the arterial is coordinated in eastbound direction during the first half of the simulation period, its westbound traffic stream is served by a PSS in the simulation's second half. Whenever a PSS is established, travel times and stops for the coordinated direction are considerably reduced (compare the simulation

period's first half in Figure 7.8a and its second half in Figure 7.8b). Travel times and stops for the less heavily travelled opposite direction are, however, increased compared to an uncoordinated signalisation (compare the simulation period's fist half in Figure 7.8b and its second half in Figure 7.8a). Considering the complete simulation period, stops are reduced by 14.5 % and 13.2 % for the east- and westbound streams, while the overall travel times remain mostly unaffected (see Table 7.4).



(a) Stream $W_N \rightarrow E_N$ (b) Stream $E_N \rightarrow W_N$

Figure 7.8: Travel time and stops for uncoordinated and coordinated signalisation (DPSS) within selected streams

Table 7.4: Reduction of travel time and stops obtained by the DPSS mechanism

	Travel time	Stops
Network	0.0 %	9.3 %
Stream $W_N \rightarrow E_N$	1.2 %	14.5 %
Stream $E_N \rightarrow W_N$	−2.1 %	13.2 %

Fuel consumption and pollution emission

In addition to travel time and stops, the fuel consumption and pollution emissions of the simulated vehicles have been evaluated. Figure 7.9 depicts fuel consumption rates for coordinated and uncoordinated signals

over the course of the simulation period. The depicted data is averaged over ten replications and shows that consumption rates can be reduced by coordination. Whenever the DPSS mechanism decreases the number of stops for the network, the fuel consumption is also decreased (compare Figures 7.7 and 7.9). Averaged over the simulation period, a reduction of 6.1 % (from 12.8 l/km 12.1 l/km) has been achieved. The rates might seem high, but can be explained the density of intersections in the network and the fact that AIMSUN's fuel consumption model does not reflect recent advances in engine development.

A decreased fuel consumption directly leads to a reduction of CO_2 emissions [198], but the emission of other pollutants can be reduced by coordination as well. For the test network, a CO reduction of 3.9 %, a HC reduction of 2.4 %, and an NO_x reduction of 7.1 % have been obtained by avoiding unnecessary stops.

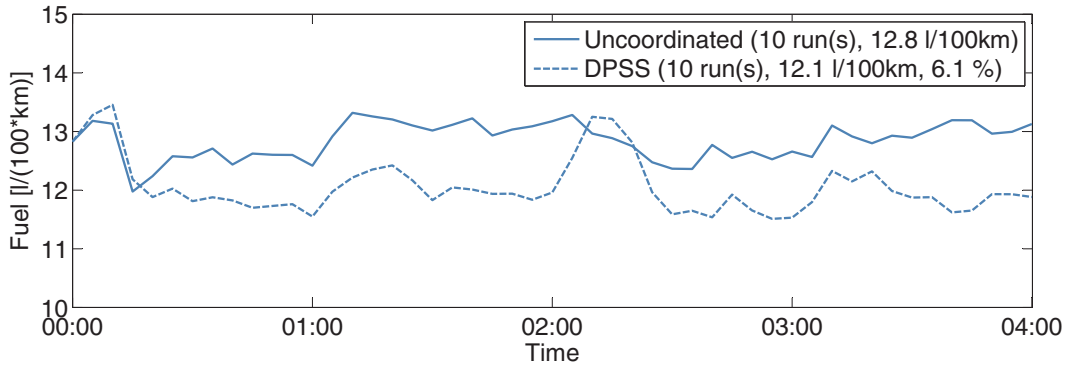


Figure 7.9: Fuel consumption for uncoordinated and coordinated signalisation (DPSS) in a Manhattan network

7.4 Summary

In urban road networks, intersections cannot optimise their signal plans independently of their neighbours. Due to the close proximity of neighbouring intersections, a coordinated signalisation is a prerequisite to avoid unnecessary stops. It can be established either implicitly (by the early detection of arriving vehicle platoons) or explicitly (by communication and offset adaptation).

With the DPSS mechanism, a decentralised coordination mechanism has been introduced. It relies on local communication to achieve a self-organised coordination of organic intersections. In response to detected traffic demands, the DPSS mechanism determines which intersections should form PSSs, negotiates a common cycle time, computes the necessary offsets, and establishes the coordination.

As signal plans are selected and optimised locally at run-time while a traffic-responsive coordination is established completely decentralised, the DPSS mechanism constitutes a novel approach. Compared to an uncoordinated organic signalisation, the DPSS mechanism can achieve a considerable reduction of stops, thereby contributing to a reduced fuel consumption and lowered pollution emissions.

Despite the promising results, the DPSS mechanism is subject to some restrictions. Due to its limited local knowledge on the network-wide demand, the mechanism constitutes a heuristic that greedily favours the strongest streams in a network. As this can impede the coordination of several other streams that in sum serve more vehicles, the resulting coordination can be suboptimal – an issue that is addressed in the following chapter.

CHAPTER 8

Hierarchical coordination of organic intersections

The DPSS mechanism that has been introduced in the previous chapter achieves a traffic-responsive coordination of organic intersections. The intersections optimise their coordinated signalisation locally. Thereby, a computationally complex and monetarily costly coordination in a traffic control centre is substituted by decentralised interactions. However, the DPSS mechanism has no information on network-wide demands such that conflicts of interest among traffic streams cannot be solved optimally. The difficulty is in selecting the coordinated streams, as the coordination of one stream can impede the coordination of several others.

Figure 8.1 illustrates an example that misleads the DPSS mechanism. The figure depicts traffic streams in a Manhattan network of six intersections. Each stream is represented by an arrow, the arrow width being proportional to the traffic flow of the particular stream. In the example, two traffic streams run from west to east, while three streams run from north to south. In sum, approximately the same amount of traffic travels in east- and southbound direction. Other traffic is neglected to keep the example simple.

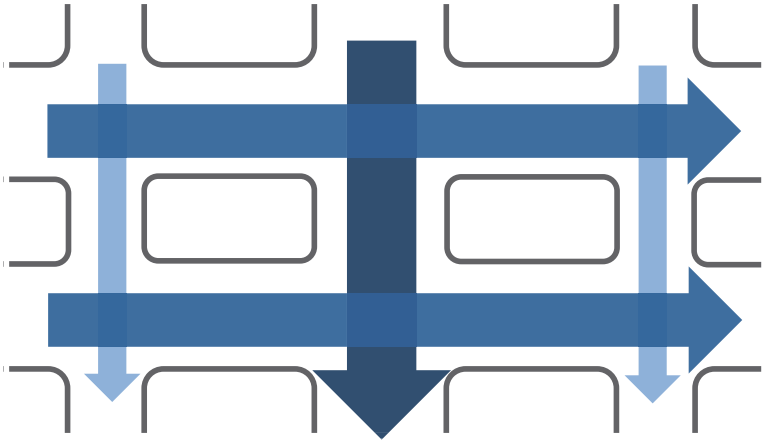


Figure 8.1: Traffic flows in a Manhattan network

For the depicted demand, the DPSS mechanism greedily establishes a PSS for the central southbound stream which is most heavily travelled. As this inhibits a coordination of both eastbound streams, further PSSs are afterwards established for the two remaining southbound streams. In consequence, vehicles travelling along one of the three southbound streams benefit from coordination when they pass the stream's second intersection without a stop. (Vehicles arrive randomly at the first intersection, compare Figure 1.4b.) However, a higher reduction of stops could be obtained by coordinating the eastbound streams. Here, travellers in each stream can pass two intersections without an additional stop such that more stops are avoided in total.

For a better treatment of similar cases, the DPSS mechanism is extended by an additional hierarchical component that is called *Regional Manager* (RM). The RM combines traffic flow measurements of several intersections and uses the resulting regional model to solve conflicts of interest among traffic streams. Thereby, it substitutes the first step of the DPSS mechanism (*Determining collaborating intersections*, see Section 7.1.1). Since the organic intersections remain responsible for their signalisation and autonomously determine the agreed cycle times and offsets for PSSs by local communication, the former decentralised system is now hierarchically organised (as illustrated in Figure 7.1c).

The remainder of this chapter is dedicated to the discussion of this hierarchical system. The presentation is partly based on a previous

publication [191]. Section 8.1 introduces the working principle of the RM and discusses its technical requirements, before the RM's relation to other hierarchical network control systems is investigated. Section 8.2 experimentally compares decentralised and hierarchical signal coordination in a simulation study to investigate possibilities and limitations of decentralised traffic control. Finally, the chapter is concluded with a summary in Section 8.3.

8.1 The Regional Manager

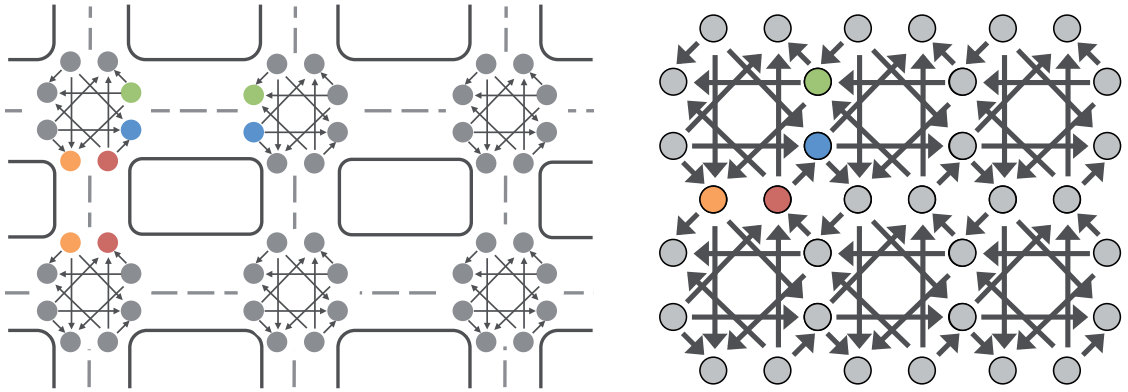
The RM's task is to determine a set of traffic streams for coordination that minimises the network-wide number of stops. To obtain a near-optimal set of streams, the RM relies on three steps. In a first step, traffic flow measurements at the organic intersections are aggregated in a graph model that represents the region's traffic flows. The regional model forms the basis for the RM's further operations, its availability is an important difference to the DPSS mechanism that relies on local demands, only. Section 8.1.1 discusses the creation of the model.

Based on the model, the RM analyses the region's traffic flows and identifies strong traffic streams that are candidates for coordination. This second step is in the focus of Section 8.1.2. Unfortunately, PSSs cannot be simultaneously installed for every candidate stream as the coordination of one stream typically inhibits the coordination of several others. Therefore, the RM combines non-conflicting streams to stream systems and selects the most promising system in a third step that is discussed in Section 8.1.3.

8.1.1 Building the network graph

To obtain a model of the regional traffic flows, the RM combines subgraphs that represent individual intersections into a single network-wide graph. Each subgraph describes the turning movements of an intersection with their traffic flows. More specifically, a subgraph contains one vertex for each outgoing section, one vertex for each incoming section, and one edge for each turning movement. Edges are directed and weighted, their weight corresponds to the current traffic flow of the represented turning.

Subgraphs are created by organic intersections which transmit them to the RM where they are connected to a regional graph by merging vertices that represent the same section. Figure 8.2a illustrates the motivational Manhattan network of six intersections with their subgraphs. In the figure, edge weights are omitted for improved readability. For the north-western intersection, adjacent sections that will be merged are individually coloured to ease their re-identification in the regional graph that is depicted in Figure 8.2b. Based on the regional demand represented by graph's edge weights (omitted in the figure), the RM determines candidate traffic streams for coordination in the following step.



(a) Network with subgraphs (b) Regional graph representation

Figure 8.2: A Manhattan network and its graph representation

8.1.2 Determining candidate traffic streams

In its second step, the RM identifies the largest traffic streams within the network. These streams are candidates for coordination as many drivers directly benefit from their coordinated signalisation.

To determine the candidate streams, the RM executes Algorithm 8.1. Using the regional graph $G = (V, E)$ created in the previous step, the algorithm iteratively connects adjacent graph edges to candidate streams after the edge set has been sorted with respect to the edge weights (Line 1).

Algorithm 8.1: Determine traffic streams

Data: A set E of edges (resulting from Step 1)
Result: A set S of streams

```
1  Sort E w. r. t. edge weights.
2  repeat
3  |  Choose edge e = arg max_weight(E).
4  |  Remove e from the edge set E.
5  |  Create a new empty stream s, add e to s.
6  |  Set edge e* := e.
7  |  repeat
8  |  |  Choose preceding edge e_p of e* with
   |  |  e_p = arg max_weight(Predecessors of e*).
9  |  |  Add e_p to stream s.
10 |  |  Set e* := e_p.
11 |  until e*'s intersection contains an edge e** with
   |  e**.weight() > e*.weight()
12 |  Repeat Lines 6–11 for the subsequent edges of e.
13 |  Remove all edges of stream s from E.
14 |  Set s.weight() to the number of benefiting vehicles.
15 |  Add stream s to set of streams S.
16 until E is empty or e.weight() ≤ threshold t.
```

When building the streams (Lines 2–16), Algorithm 8.1 chooses an unprocessed edge e with maximum weight from the edge set E (Line 3), removes e from E (Line 4), and adds e to the traffic stream s that is currently under construction (Line 5).

Starting from the first turning e^* in the stream (that is currently $e^* = e$, see Line 6), the algorithm iteratively determines the best predecessor turning by selecting the adjacent edge with highest weight from the particular candidates (Lines 7–11). The selected edge e_p is included in the stream s (Line 9) as predecessor of e^* and the iteration continues with $e^* = e_p$ as the stream's first turning. The addition of predecessors is stopped if e^* does not represent the most heavily travelled turning at its intersection (Line 11), as in this case the corresponding intersection

prefers a different coordination. Once the predecessor turnings have been added to the stream, successor turnings are analogously included by starting from the stream's last turning (Line 12). Afterwards, the stream s is completed.

Before the completed stream is included in the set S of candidate streams (Line 15), its edges are removed from the edge set E (Line 13) as their represented turnings cannot be included in another coordinated stream. Furthermore, a weight is assigned to the stream (Line 14). The weight estimates the number of vehicles that benefit from the stream's coordination. For a stream $s = (e_1, \ldots, e_n)$ that incorporates n turnings, it is estimated as

$$s.weight() := \sum_{i=2}^{n} e_i.weight(),$$

i. e., vehicles are assumed to benefit from a coordinated signalisation once they have passed the stream's first signalised turning, where they arrive randomly.

Once the completed stream s has been added to the stream set, the construction of streams continues for the reduced edge set E until the remaning edges do no longer serve considerably large traffic flows.

To illustrate the stream set S resulting from Algorithm 8.1, Figure 8.3 depicts selected streams for the Manhattan network that has served as motivational example. The figure shows constructed streams as connected sequences of red edges in the network's regional graph. Some streams in S are conflicting, i. e., they cannot be coordinated simultaneously. Before the selection of conflict-free subsets will be discussed, the run-time of Algorithm 8.1 is analysed.

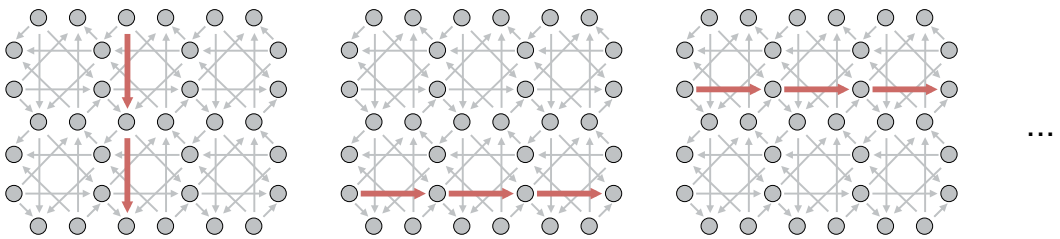


Figure 8.3: Candidate traffic streams in a Manhattan network

Algorithm 8.1 has a run-time complexity of $O(m \log(m))$ with m being the number of edges in the regional graph. The time is required for sorting (Line 1). During the execution of Lines 2–16, every edge is looked at a constant number of times, only. Edges included in a stream are removed (Lines 4 and 13) and will not be considered in later iterations. The repeated execution of Lines 6–11 and 12 can be performed in linear time using a suitable data structure, since the degree of vertices is bounded by a constant when representing intersections. Furthermore, since each edge is part of at most one stream, evaluating the benefits of all constructed streams (Line 14) can be done in linear time.

The complexity of Algorithm 8.1 can be denoted in the same way depending on the number of intersections, since the number of turnings – which are represented as edges in the graph – depends linearly on the number of intersections. In consequence, Algorithm 8.1 constitutes a computationally efficient heuristic that identifies a set of strong traffic streams in a road network. The set serves as input for the third step of the regional coordination.

8.1.3 Determining stream systems

In a third step, the RM determines subsets of non-conflicting streams (so-called *stream systems*). Traffic streams in the same stream system must not intersect each other or run in different directions on the same roads. In other words, each signalised intersection can be part of at most one stream to guarantee that the streams within a stream system can be coordinated simultaneously. Among the identified stream systems, the RM determines one system that minimises the network-wide number of stops. This system will be implemented in the road network.

Figure 8.4 illustrates three stream systems that could have been obtained for the Manhattan network example. As before, streams are depicted as connected sequences of red edges. Each of the depicted stream systems is non-conflicting. However, the systems differ in their savings (that depend on the network's traffic flows and are not shown in the figure).

To create promising stream systems, the RM executes Algorithm 8.2 on the previously identified set S of traffic streams. The algorithm identifies non-conflicting subsets of streams without generating the power set of S.

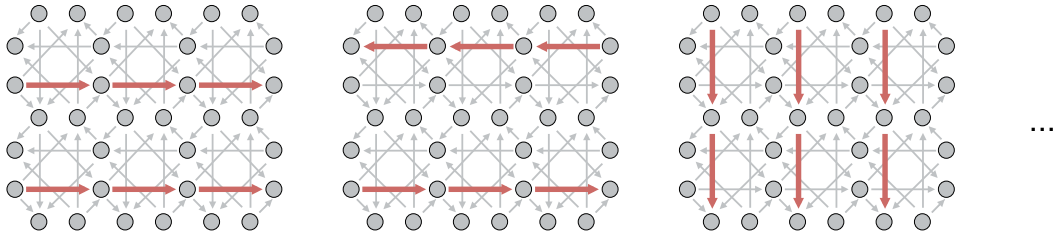


Figure 8.4: Candidate stream systems in a Manhattan network

In a preprocessing step (Lines 1–8), an incompatibility function $J\colon S \times S \to \{0,1\}$ is constructed that allows to quickly identify conflicting streams in later stages of the algorithm. For each intersection i, traffic streams that include one of i's turnings are identified (Line 4). Pairs of these streams are conflicting and are thus marked correspondingly (Lines 5–7). Once the preprocessing is completed for all intersections, the constructed function J indicates the (in)compatibility of arbitrary pairs (s, s') of streams in S. A value of $J(s, s') = 0$ indicates a non-conflicting pair of streams, while a value of $J(s, s') = 1$ indicates a conflict. It should be noted that $J(s, s) = 1$ for each stream $s \in S$.

Using the function J, stream systems are iteratively created by greedily including the best non-conflicting and unprocessed traffic streams that are available. To quickly access the best streams throughout the iterative process, S is sorted with respect to the streams' weights (Line 9). A set U contains the unprocessed streams. Initially, it is a copy of S (Line 10). The algorithm terminates when each stream in S has been included in a stream system, i. e., the algorithm terminates when U is empty (Line 29).

To obtain non-conflicting stream systems, Algorithm 8.2 relies on two nested loops. In every iteration of the outer loop (Lines 11–29), a stream system Z is created. A set T keeps track of streams that are not in conflict with any stream in Z. Initially, Z is empty such that T equals S (Lines 12 and 13). In Lines 14 and 15, the best unprocessed stream s_1 is added to Z. After its inclusion, s_1 is removed from the set U of unprocessed streams (Line 16). Streams that are in conflict with s_1 are removed from T (Lines 17–19). Here, the incompatibility function J from preprocessing helps to identify conflicting streams.

After the inclusion of the first stream, the inner loop in Lines 20–27 is processed. The loop iterates over the set T and determines the best non-

conflicting stream that can be added to the stream system Z (Line 21). The stream is added to Z (Line 22) and can be removed from U as it has been processed (Line 23). Subsequently, the set T is updated by removing streams in conflict with the newest addition to Z (Lines 24–26). The inner loop ends when T is empty, i. e., when no non-conflicting streams are available for addition. Now, Z is completed and can be added to the set $\mathcal{R}$ that collects the created stream systems.

Using suitable data structures, Algorithm 8.2 has a complexity of $O(m^2)$, with m being equal to the number of edges in the regional graph. The preprocessing (Lines 1–8) needs time $O(m^2)$, since the number r of streams containing the same intersection is bounded by a constant (the intersection's number of turnings). Furthermore, at most $O(m)$ entries of the function J are set to 1 during preprocessing. Therefore, the amortised complexity of the inner loop in Lines 20–27 is $O(m)$, and the loop in Lines 11–29 can be executed in time $O(m^2)$.

The result of Algorithm 8.2 is a set $\mathcal{R}$ of stream systems. To determine the benefiting vehicles for a stream system $Z \in \mathcal{R}$, the number of vehicles benefiting from the coordination of each stream $s \in Z$ is summed up, i. e.,

$$Z.weight() := \sum_{s \in Z} s.weight().$$

Based on the obtained values, the best stream system is selected for implementation in the road network. The RM communicates the resulting partnerships to the intersections where a common cycle time, appropriate signal plans, and offsets for the PSSs are determined locally using Steps 2 and 3 of the DPSS mechanism (see Sections 7.1.2 and 7.1.3).

8.1.4 Discussion

The RM constitutes an extension to the DPSS mechanism that supports the coordination of signalised intersections in a road network. The remainder of this section will discuss the RM's technical requirements, before it focuses on the remaining limitations of hierarchical control. Finally, the RM will be situated in the context of other network control systems.

Algorithm 8.2: Determine stream systems

Data: A set S of streams (determined during Step 2)
Result: A set $\mathcal{R}$ of generated stream systems

Construct a function $J\colon S \times S \to \{0, 1\}$.
Set J to 0 for all arguments.
foreach *Intersection* i **do**
    Determine the streams $s_{i,1}, \ldots, s_{i,r}$ containing i.
    foreach $j, k \in \{1, \ldots, r\}$ **do**
        Set $J(s_{i,j}, s_{i,k}) := 1$.
    end
end
Sort S w. r. t. the number of benefitting vehicles.
Set $U := S$.
repeat
    Set $T := S$.
    Create an empty stream system Z.
    Choose stream s_1 with $s_1 = \arg\max_{weight}(U)$.
    Add s_1 to Z.
    Remove s_1 from U.
    foreach *Stream* s' *with* $J(s_1, s') = 1$ **do**
        Remove s' from T.
    end
    repeat
        Choose stream s_2 with $s_2 = \arg\max_{weight}(T)$.
        Add s_2 to Z.
        Remove s_2 from U.
        foreach *Stream* s' *with* $J(s_2, s') = 1$ **do**
            Remove s' from T.
        end
    until T *is empty.*
    Add Z to $\mathcal{R}$.
until U *is empty.*

Requirements

Since the RM extends the DPSS mechanism, it has the same minimum requirements regarding traffic detection, communication, and time synchronisation that are needed for decentralised control (see Section 7.1.5).

Additionally, a regional control centre is required such that the RM can select the coordinated traffic streams. To build the necessary regional model (Step 1, see Section 8.1.1), the intersections need to communicate their traffic flow measurements to the RM. Thus, they have to be connected to the control centre via a communication link. The link is also required during Step 3 (see Section 8.1.3) as the RM has to inform the intersections on the coordinated streams.

Both steps are periodically executed in intervals of a few minutes. The amount of communicated data is limited and delays of several seconds during data transmission are not critical. In consequence, the RM has no special requirements regarding communication bandwidth and latencies.

Regarding the RM's computational requirements, a standard PC is sufficient for the on-line coordination of typical traffic networks that consist of several dozen signalised intersections. The computationally expensive Steps 2 and 3 can be implemented efficiently with respect to the number of intersections in the network (see Sections 8.1.2 and 8.1.3 for a run-time analysis).

In summary, the technical requirements for the RM are moderate and should be justified by the benefits of a hierarchically coordinated signalisation (see Section 8.2). Nevertheless, the RM is still subject to some limitations.

Limitations

Despite its regional model, the RM can be misled by virtual streams. The regional graph is based on flow measurements at the signalised intersections, but the routes of the vehicles are a-priori unknown and have to be derived from the model. In special constellations, the traffic streams identified by Algorithm 8.1 may not coincide with the actual streams in the network. A corresponding problem has been observed for the DPSS mechanism and is discussed in Section 7.1.5.

A second limitation of the RM results from the complexity of the optimisation problem and the run-time restrictions prevalent in an on-

line system. To allow for an efficient implementation, the RM identifies traffic streams and stream systems by a greedy approach. Thus, the stream system selected for coordination is not guaranteed to be optimal. However, as changing traffic flows and random fluctuations affect the performance of a coordinated signalisation, near-optimal (but up to date) solutions are preferred over optimal solutions for outdated traffic demands.

Comparison

Despite its limitations, the RM has several advantageous properties that become clear when the approach is situated in the context of centralised and hierarchical network control systems.

In contrast to centralised systems (like SCOOT, see Section 2.4.1), the organic intersections remain autonomous entities even when their coordination is supported by the RM. The intersections continue the on-line optimisation of their signal plans and can negotiate coordination updates (like a changed agreed cycle time) without any involvement of the RM.

The limited dependency on the RM has two important advantages compared to centralised systems: Firstly, the RM does not constitute a single point of failure. The DPSS mechanism remains functional without the RM even though the performance of the coordination can suffer (*graceful degradation*). In centralised systems, a network's signalisation is widely dependent on a functional control centre.

The second advantage affects the communication among intersections and control centre. As the RM is active periodically in intervals of several minutes, only, and as the amount of transmitted data is limited, the approach works for unreliable or slow communication links with limited bandwidth. In centralised systems, where detection and signalisation data has to be continuously transmitted among control centre and intersections, the communication links are vital, but often error-prone [74].

Considering its architecture, the RM is more closely related to hierarchical systems like BALANCE or MOTION (see Section 2.4.4). Both systems rely on a control centre to optimise frame signal plans for the network's intersections, but allow for local traffic-actuated adaptations.

Compared to these systems, the RM puts an even stronger emphasis on distributed intelligence. Instead of a merely traffic-actuated operation, the organic intersections evaluate and optimise their signalisation on-line. Unfortunately, both philosophies could not be compared experimentally for this thesis as neither an AIMSUN implementation of BALANCE or MOTION nor reference networks have been available. The following experimental evaluation will therefore focus on a comparison of the DPSS mechanism with and without the support of the RM.

8.2 Experimental evaluation

To study the potential benefits of the RM that supports the signal coordination with data on the regional traffic demand, a decentralised coordination (by the DPSS mechanism) and a hierarchical coordination (supported by the RM) have been compared in a simulation study. The experimental setup of the study is presented in Section 8.2.1. Then, Section 8.2.2 discusses the obtained results.

8.2.1 Test case

In the simulation study, the Manhattan network that served as test case for the evaluation of the DPSS mechanism has been revisited (see Figure 7.6). The network consists of six intersections that each support twelve turning movements. The connecting road segments have a length of 250 m to 350 m, are two-laned, and provide an additional side-lane for left-turns. Each intersection is equipped with an observer/controller responsible for the traffic-responsive reconfiguration of a fixed-time signal plan. The plan combines the intersection's eight signal groups in four phases. Phases 1 and 3 serve left-turning vehicles, while Phases 2 and 4 serve vehicles going straight ahead or turning right.

The simulated traffic demand has been designed to resemble the motivational example that misleads the DPSS mechanism (see Figure 8.1). Table 8.1 lists the simulated demands for all O/D pairs in the network. During the first half of the four hour simulation period, the most relevant traffic streams are south- and eastbound. The strongest stream starts at origin N and ends at destination S (see Figure 7.6 for O/D labels), but

Table 8.1: Traffic demands for the Manhattan network

Traffic demands for O/D pairs (in veh/h)			1st half →	1st half ←	2nd half →	2nd half ←
W_N	$\leftrightarrow$	E_N	775	345	345	775
W_S	$\leftrightarrow$	E_S	775	345	345	775
N_W	$\leftrightarrow$	S_W	430	175	175	430
N	$\leftrightarrow$	S	1035	345	345	1035
N_E	$\leftrightarrow$	S_E	430	175	175	430
Others			10	10	10	10
Total			5630		5630	

both eastbound streams (starting at origins W_N and W_S, respectively) are also heavily travelled. In the second half of the simulation period, all traffic flows change their direction such that most traffic is north- and westbound.

Using the simulation model as test case, the DPSS mechanism with and without the RM as hierarchical extension is compared to an uncoordinated organic signalisation that serves as reference scenario. The intersection's observer/controller components are configured as in earlier studies. The applied levels for all design factors are available in Table 7.1. For coordination, the DPSS mechanism uses a check frequency of five minutes, an agreed cycle time tolerance of $ACT_{tol} = 2\,\text{s}$, and a signal plan tolerance of $sp_{tol} = 10\,\text{s}$. All settings correspond to recommendations of earlier sensitivity studies.

The comparison of the decentralised and hierarchical coordination considers travel times and stops for the network and for selected traffic streams as response variables. Additionally, fuel consumption and pollution emissions are evaluated and compared. To account for the stochasticity of the experimental setup, all comparisons are based on the average of ten replicated simulation runs with different random seeds. The necessary simulations have been conducted using AIMSUN v. 5.1.11 running under Microsoft Windows Vista 64-bit on a 2.5 GHz Intel Core 2 Quad processor equipped with 8 GB RAM.

8.2.2 Simulation results

The test network has been simulated for coordinated and uncoordinated organic intersections. In the coordinated case, intersections apply the DPSS mechanism either with or without the RM supporting the coordination. When not supported by the RM, the DPSS mechanism creates a PSS serving the traffic stream from origin N to destination S during the first half of the simulation period. As this inhibits a coordination of both eastbound streams, further PSSs are afterwards established for the two remaining southbound streams. In contrast, both eastbound streams are coordinated when the DPSS mechanism is supported by the RM.

When the traffic demand changes at the beginning of the simulation's second half, the signal coordination needs to be adapted. During this period, the DPSS mechanism coordinates the three northbound streams, while the RM establishes PSSs for the two westbound streams.

Network-wide travel times and stops

The different PSSs that are established by the DPSS mechanism when it is operated with and without the RM affect the performance of the signalisation. Figure 8.5 depicts the network-wide travel times and stops resulting from the different coordination schemes. The depicted data is averaged over ten replications. Error bars are omitted for improved readability.

Compared to the uncoordinated reference solution, both coordination mechanisms can reduce the number of stops for nearly the complete simulation period. The abrupt change of the traffic demand in the beginning of the simulation's second half temporarily affects their performance, but once the change is detected, the coordination is updated and the stops are reduced to their previous levels. As expected for the investigated test case, the DPSS mechanism achieves the highest reduction of stops when it is supported by the RM. Over the simulation period, 15.7 % of the network-wide stops are saved in comparison to the reference solution. Without the RM, the DPSS mechanism achieves a reduction of 7.6 %.

Despite the reduction of stops, the network-wide travel times are mostly unaffected by coordination. Without the RM, travel times are reduced by 2.4 % compared to the uncoordinated reference. With the RM, a slight

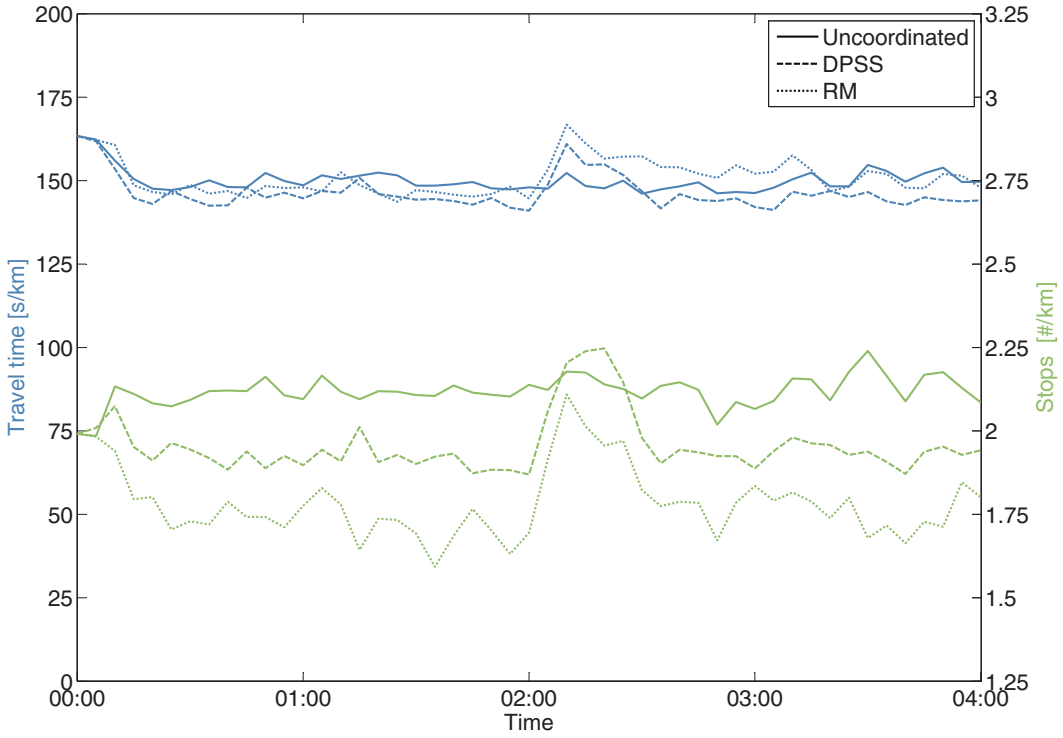


Figure 8.5: Travel time and stops for uncoordinated and coordinated signalisation (DPSS/RM) in a Manhattan network

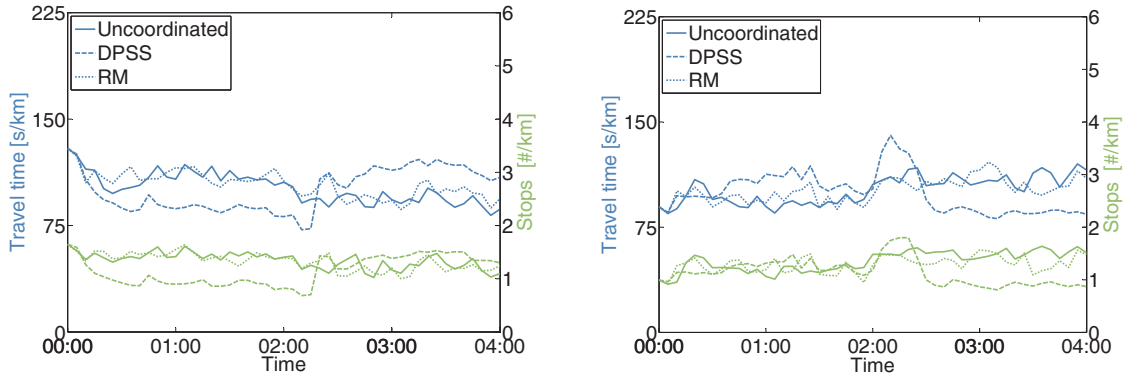
increase of 0.6 % has been observed. Due to long cycle lengths and a reduced flexibility of the coordinated intersections, travel time savings resulting from a reduced number of stops for the coordinated phases are partly compensated. This effect has been observed in earlier experiments and is discussed in Section 7.3.2.

Travel times and stops for selected traffic streams

To illustrate the different effects of decentralised and hierarchical coordination, selected traffic streams have been evaluated with respect to travel times and stops.

North- and southbound streams Figure 8.6 compares the travel times and stops that have been measured for the Streams $N \rightarrow S$ and $S \rightarrow N$. Without the support of the RM, the DPSS mechanism establishes a PSS for Stream $N \rightarrow S$ during the first half of the simulation. During this

period, the stream is the most heavily travelled stream in the network. Its coordination successfully reduces travel times and stops for southbound travellers (see Figure 8.6a), but increases both measures for the less heavily travelled opposite direction (see Figure 8.6b).



(a) Stream $N \rightarrow S$ (b) Stream $S \rightarrow N$

Figure 8.6: Travel time and stops for the uncoordinated and coordinated signalisation (DPSS/RM) of selected north- and southbound streams

During the simulation's second half, the situation is reversed due to the changed traffic demand. The DPSS mechanism establishes a PSS for Stream $S \rightarrow N$ which is most heavily travelled after the change. The PSS reduces the travel time and stops for northbound travellers (see Figure 8.6b), but the reduction comes at the cost of increased values for the less heavily travelled opposite direction (see Figure 8.6a).

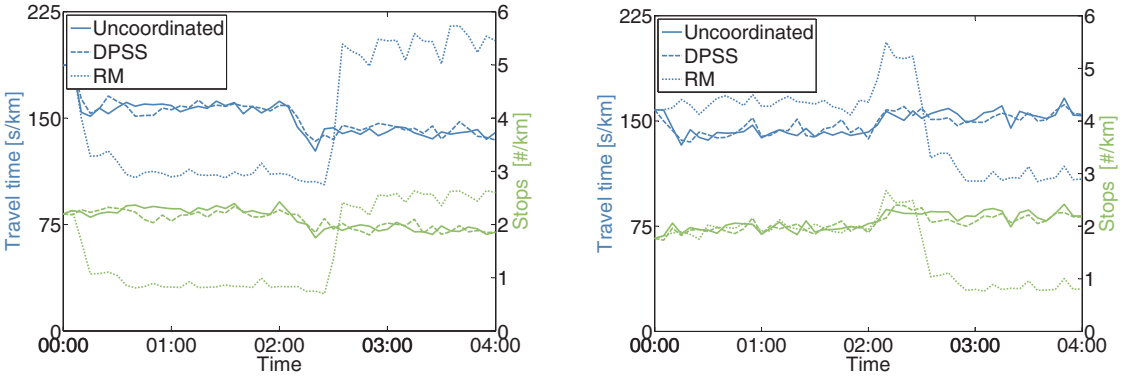
Considering the complete simulation period, the DPSS mechanism reduces travel times and stops for both streams. Table 8.2 summarises the reductions compared to an uncoordinated signalisation.

In contrast to a decentralised coordination, the RM does not establish PSSs for the north- or southbound streams. In consequence, measured travel times and stops for these streams approximately resemble those observed for an uncoordinated signalisation (see Figure 8.6 and Table 8.2). The network-wide benefits of an hierarchical coordination result from the coordination of the east- and westbound streams.

Table 8.2: Reduction of travel time and stops compared to an uncoordinated signalisation of selected north- and southbound streams

	Travel time		Stops	
	$N \rightarrow S$	$S \rightarrow N$	$N \rightarrow S$	$S \rightarrow N$
DPSS	1.3 %	2.3 %	14.4 %	15.6 %
RM	−2.5 %	0.4 %	1.4 %	4.4 %

East- and westbound streams Figure 8.7 depicts results obtained for the Streams $W_N \rightarrow E_N$ and $E_N \rightarrow W_N$. Without the support of the RM, the DPSS mechanism does not establish PSSs for these streams. In consequence, the measured travel times and stops approximately resemble those obtained by an uncoordinated signalisation.



(a) Stream $W_N \rightarrow E_N$ (b) Stream $E_N \rightarrow W_N$

Figure 8.7: Travel time and stops for the uncoordinated and coordinated signalisation (DPSS/RM) of selected west- and eastbound streams

The RM coordinates the eastbound stream during the first half of the simulation period, while the opposite direction is served by a PSS during the simulation's second half. Whenever a PSS is established, travel times and stops for the coordinated direction are considerably reduced (compare the simulation period's first half in Figure 8.7a and its second half in Figure 8.7b). Travel times and stops for the less heavily travelled opposite direction are, however, increased compared

to an uncoordinated signalisation (compare the simulation period's fist half in Figure 8.7b and its second half in Figure 8.7a). Considering the complete simulation period, stops are reduced by 26.5 % and 23.6 % for the east- and westbound streams, while the overall travel times remain mostly unaffected. All results are summarised in Table 8.3.

Table 8.3: Reduction of travel time and stops compared to an uncoordinated signalisation of selected east- and westbound streams

	Travel time		Stops	
	$W_N \rightarrow E_N$	$E_N \rightarrow W_N$	$W_N \rightarrow E_N$	$E_N \rightarrow W_N$
DPSS	−0.6 %	0.0 %	1.1 %	1.7 %
RM	0.6 %	0.6 %	26.5 %	23.6 %

Fuel consumption and pollution emission

To assess the environmental impact of decentralised and hierarchical signal coordination, AIMSUN's environmental models have been used to evaluate the vehicles' fuel consumption and their emission of pollutants. The models have been configured based on data taken from [67, 149, 197] (see Section 7.3.1 for details).

Figure 8.8 depicts the fuel consumption over the simulation period and shows that both coordination mechanisms reduce the fuel consumption compared to an uncoordinated signalisation. The highest reduction of 1.2 l/100km (or 8.9 %) is obtained by the RM. Without the RM's support, the DPSS mechanism still achieves a considerable reduction of 0.5 l/100km (or 3.8 %).

The temporal development of the fuel consumption closely resembles the development of the network-wide number of stops (compare Figures 8.5 and 8.8). Once a coordinated signalisation is established in the beginning of the simulation period, stops and, in consequence, fuel consumption rates are reduced. After the change in traffic demand in the beginning of the simulation's second half, both measures are temporarily increased until the coordination is adapted to the new demand. The close relation is due to the additional fuel that is required to re-accelerate a stopped vehicle. Therefore, a network-wide reduction of stops does not only improve the

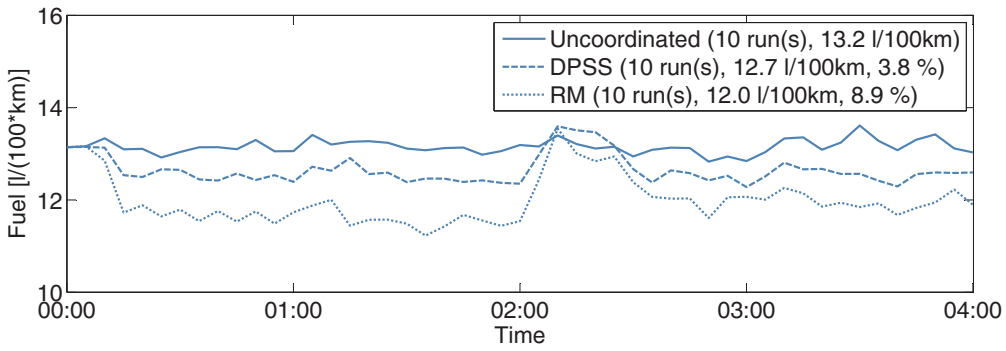


Figure 8.8: Fuel consumption for uncoordinated and coordinated signalisation (DPSS/RM) in a Manhattan network

comfort for the drivers, but is also beneficial for environmental reasons.

The reductions reported for fuel consumption directly map to the emission of Carbon Dioxide (CO_2) that is proportional to the quantity of fuel consumed [198]. The emission of other pollutants is summarised in Table 8.4. The decentralised and the hierarchical coordination are both able to reduce the emission of pollutants compared to an uncoordinated signal system, but better results with respect to all pollutants are obtained by the RM. Again, the higher reduction of stops achieved by a supporting RM explains the higher reduction of emissions. CO, NO_x, and HC are emitted especially during high load and idling periods of petrol and diesel engines, i. e., when vehicles are standing with running engines or when they have to accelerate after a stop. These periods are reduced by coordination.

Table 8.4: Network-wide reduction of pollution emissions compared to an uncoordinated operation

	CO	NO_x	HC
DPSS	2.8 %	4.2 %	2.3 %
RM	5.2 %	9.8 %	2.9 %

In summary, both presented traffic-adaptive coordination mechanisms have a beneficial impact on the traffic network. Especially, the number of stops, the fuel consumption, and pollution emissions can be reduced by coordination. For the investigated test case – that exploits a weakness

of the DPSS mechanism – the hierarchical coordination shows a better performance.

8.3 Summary

The RM constitutes an extension to the DPSS mechanism that can further improve the coordination of traffic signals. With the RM's support, the optimisation of a network's signalisation is tackled hierarchically. On a regional level, the RM determines traffic streams for coordination. The corresponding PSSs are subsequently established by the organic intersections with the help of local interactions.

The hierarchical problem decomposition allows for a better consideration of conflicts of interest among traffic streams. To obtain the necessary regional view on the network's demands, the RM combines the intersections' flow measurements in a regional model. Based on the model, strong traffic streams in the network are identified and combined into a non-conflicting stream system that, when coordinated, minimises the network-wide number of stops.

While a regional model is a common property of centralised or hierarchical network control systems, the hierarchical extension of the DPSS mechanism differs from these systems by its strong emphasis of local on-line optimisation. The RM is solely responsible for the selection of the coordinated traffic streams, as this is a task that strongly benefits from a regional model. The relevant coordination parameters are, however, determined in a decentralised process among the organic intersections. The intersections also remain responsible for the optimisation of their own signal plans. This constitutes a difference even to existing hierarchical systems like BALANCE or MOTION (see Section 2.4.4) that allow for local traffic-actuated adaptations within the close boundaries of a frame signal plan, only.

CHAPTER 9

Conclusion

Motivated by the vision of adaptive learning intersections that collaborate to optimise their signalisation, this thesis has introduced a self-organising observer/controller framework for signal control. This chapter concludes the work by summarising its major contributions in Section 9.1. Section 9.2 collects remaining open issues and outlines directions for future research, before the thesis is closed with final remarks in Section 9.3.

9.1 Summary

To get closer to the idea of autonomous self-organising intersections, the thesis progressed in several steps. The initial step focused on a single signalised intersection. With the help of the generic observer/controller architecture [23, 157, 159], the intersection has been endowed with the ability to evaluate and optimise its signalisation at run-time. The step and its contributions to traffic signal control are summarised in Section 9.1.1.

At an intersection, it is crucial to ensure that a learning observer/controller does not negatively affect the signalisation by learning errors. The

issue has been addressed by the two-levelled learning mechanism that is embodied in the generic observer/controller architecture. By combining on-line reinforcement learning and simulation-based off-line optimisation, this thesis has presented the first successful implementation of two-levelled learning for the observer/controller framework. The mechanism and its implications for Organic Computing and machine learning are recapitulated in Section 9.1.2

Once the single intersections have become autonomous entities, the focus of the thesis has been broadened to road networks. In networks, the traffic-responsive coordination of signals is an important aspect. Following the trend towards distributed intelligence, the thesis introduced a self-organising coordination mechanism. The mechanism relies on locally collaborating intersections and works completely decentralised. However, it can be converted into a hierarchical system by the addition of a Regional Manger. The manager resolves conflicts of interest among traffic streams based on a network-wide traffic model, thereby supporting the coordination process. Both variants of the coordination mechanism and their contributions to traffic signal control are summarised in Section 9.1.3.

9.1.1 Organic intersections

The signalisation of today's fixed-time or traffic-actuated signal controls is widely predetermined at design time. A fixed-time signal plan is configured to suit an expected average traffic demand that is derived from past experience. Changes in demand are handled by time-dependent schedules that switch among a small number of precalculated plans. In comparison, traffic-actuated controls are more flexible and can adapt their signalisation. Their behaviour is determined by predefined temporal and logical conditions, though. At run-time, no further evaluation or optimisation of the signalisation takes place.

Both fixed-time and traffic-actuated controls are not well prepared to handle unexpected traffic demands that can be caused by road works or incidents in the network or that can result from public events like sport matches, concerts, or strikes. The handling of such events requires to shift the optimisation of signal plans from the design time to the run-time of the signal system. To this end, the thesis adapted the

generic observer/controller framework proposed for Organic Computing [23, 157, 159] to signal control. A fixed-time or traffic-actuated signal controller is extended by an observer/controller that monitors the local traffic flows, evaluates the performance of the active signal plan, and adapts the signalisation when necessary. The adaptation relies on a two-levelled learning mechanism that combines on-line reinforcement learning and off-line simulation-based optimisation.

The optimisation of an intersection's signalisation at run-time is not only advantageous in the presence of special events, but can also be beneficial in the more frequent case of regularly reoccurring traffic demands. To this end, two intersections at Hamburg, Germany, have been investigated in a simulation study. The study considered the typical work-day traffic at both intersections and compared their time-dependent field signal plans to the signalisation of the observer/controller framework. Results show that the observer/controller considerably reduces the average vehicular delay throughout the day.

A price to pay for the obtained reduction are increased computational requirements for the selection and optimisation of signal plans. Since evolutionary optimisations are particularly demanding, a study has investigated the use of approximation- and simulation-based fitness functions. Approximation-based signal plan evaluations are relatively fast, but not as flexible as traffic simulations. Since simulations are computationally demanding and noisy, special care is required when configuring the simulation environment. To this end, the thesis has investigated strategies for the handling of simulated durations, random seeds, and reevaluations throughout an evolutionary search. Assuming a limited time budget, the best found strategy improves the quality of the evolved signal plans considerably. Although the concrete findings are simulator-specific, insights on the advantage of a fixed random seed and the benefit of accurate evaluations in the final generations of a run should carry over to other noisy problems as well.

Despite the careful investigations, the observer/controller framework requires the computational power of an embedded PC that needs to be connected to the intersection's signal controller. A real-world evaluation of the framework would thus require a close collaboration with a manufacturer of signal controls and is beyond the scope of this thesis.

However, simulations prove the principal applicability of the proposed observer/controller for on-line signal plan optimisation. The obtained delay reductions are expected to carry over to a real-world setting.

9.1.2 Two-levelled learning

The observer/controller's two-levelled learning mechanism combines a *Learning Classifier System* (LCS) for on-line signal plan selection with an *Evolutionary Algorithm* (EA) for off-line signal plan optimisation. As rule-based reinforcement learning system, the LCS learns a mapping of traffic demands to signal plans. The mapping is continuously revised and updated based on reinforcement from the signalised intersection, while optimised signal plans for unknown demands are provided by the EA. The EA's search is based on a model of the controlled intersection such that candidate signal plans can be evaluated without a negative influence on the real intersection that forms the *System under Observation and Control* (SuOC).

The combination of reinforcement learning and optimisation addresses several weaknesses of the individual mechanisms: The first issue is related to the run-time requirements of EAs. Depending on the evaluation function, evolutionary optimisations can be time-consuming which hinders their use in many on-line applications. Here, the on-line optimisation of traffic signals is a prominent example that has only been addressed recently (see Section 3.1.3). Although two-levelled learning cannot speed up the optimisation process, the LCS provides an intelligent memory that stores and adapts the optimisation results. Thereby, EAs become applicable in scenarios where their time requirements would otherwise be problematic.

In contrast to EAs, Michigan-style LCSs have been designed with on-line applications in mind. However, they are known to require numerous learning interactions before they can satisfyingly solve a complex task. This is problematic in scenarios where environmental reinforcement is scarce. Considering a signalised intersection, the vehicular delay of a signal plan is available only after the plan has been active for at least one cycle, i. e., typically after one minute or later. Since the problem exhibits numerous potential actions, the situation gets even worse. Two-levelled

learning addresses this issue by an additional EA that supports the LCS. The EA determines suitable actions off-line based on a model. Thus, the task of the LCS is reduced to learn correct classifier conditions for the provided actions and to correct model errors based on environmental reinforcement.

A second issue that is simultaneously addressed by the model-based optimisation of signal plans is the exploration-exploitation dilemma [186]. A learning mechanism can either exploit its previously learnt knowledge and perform the best known action or it can explore a new alternative to get better over time. The risk of exploration is that most explored actions will have a detrimental outcome which is problematic for the productive system. With two-levelled learning, action exploration is performed by the EA based on a model. Thus, negative effects on the productive system are avoided as the LCS can exploit relatively good situation-action mappings that will be adapted based on reinforcement when necessary.

In this thesis, two-levelled learning in the observer/controller has been successfully applied for the first time. Although the implementation contains some problem-specific components (like the widening mechanism with its capacity check, see Section 5.4.1), two-levelled learning can be adapted to a wide area of safety- and performance critical applications that have been inept for on-line optimisation and learning before. Recently, the two-levelled mechanism has been applied for the optimisation of protocol parameters in communication networks [193].

9.1.3 Self-organised coordination

With the help of the observer/controller framework and its two-levelled learning mechanism, a signalised intersection autonomously optimises its signalisation at run-time. In a road network, interdependencies among neighbouring intersections need to be considered, though. A coordinated signalisation (that results in *progressive signal systems* or *green waves*) is important to reduce the number of stops and, in consequence, travel times, fuel consumption, and pollution emissions.

To allow for the traffic-responsive coordination of organic intersections, the self-organising *DPSS mechanism* has been introduced. Neighbouring

intersections communicate their traffic flows, decide on coordination partnerships, negotiate a common cycle time, and establish a coordinated signalisation by adapting their offsets accordingly. The DPSS mechanism works decentralised and achieves a considerable reduction of stops compared to an organic, but uncoordinated signalisation. Compared to other self-organising coordination mechanisms like those discussed in Section 3.3, the DPSS mechanism does not require sophisticated sensor equipment for the early detection of arriving vehicle platoons and is not restricted to a limited number of signal plans to choose from.

Due to its decentralised working principle, the DPSS mechanism can be misled to establish a suboptimal coordination for some traffic demands, though. To address this issue, a hierarchical extension – the *Regional Manager* (RM) – has been proposed. The RM collects data on the traffic flows at the organic intersections and creates a regional traffic model that is used to identify a set of non-conflicting traffic streams for coordination. The selected streams are subsequently coordinated by the DPSS mechanism. The coordinated intersections remain autonomous entities and continue to optimise their signalisation locally.

In situations where the DPSS mechanism is misled by a network's traffic demands, a supporting RM can resolve conflicts of interest among the traffic streams such that a better reduction of stops is achieved. Unlike most centralised network control systems, the RM does not require a continuous and often error-prone transmission of detection and signalisation data among control centre and intersections. Moreover, the strong emphasis on distributed intelligence and local optimisation distinguishes the RM from other hierarchical control systems that merely allow for local traffic-actuated operations. Thereby, the DPSS mechanism and its regional extension contribute to the state of the art in adaptive network control.

9.2 Outlook

With the introduction of an observer/controller that autonomously adapts an intersection's signalisation by two-levelled learning and that moreover supports the self-organised coordination of organic intersections, the objectives of this thesis have been successfully addressed. However, as stated by Müller-Schloer et al. [138]

> "[...] good research answers some questions while – and this is more important – posing new ones."

This outlook discusses new questions and ideas for future research that arise from the insights gained in this thesis. Application-specific aspects are discussed in Section 9.2.1, while ideas that are more generally related to the observer/controller architecture are summarised in Section 9.2.2. Finally, Section 9.2.3 explores the potentials of collaborative learning and self-organisation in road networks.

9.2.1 Traffic signal control

To further improve the organic intersections proposed in this thesis, three extensions are considered most promising. Firstly, the support of *long-term predictions* based on learnt historical time series would enable intersections to proactively adapt their signalisation with respect to expected developments. Thereby, the handling of reoccurring changes (like peak hours) can be improved. The required forecast techniques are well established [47], their incorporation in the observer/controller framework does not pose a conceptual challenge [205].

A further improvement should be obtainable by incorporating all *basic signalisation parameters* in the optimisation process. In the conducted experiments, the phase durations, the cycle time, and – in the coordinated case – the offset of a signalised intersection have been considered. By utilising a predefined set of allowable phase transitions, the phase sequence can be incorporated as additional parameter that allows to minimise the green times which are lost during phase transitions. Thereby, the vehicular delay at a signalised intersection can be further reduced, but road users have to get accustomed to the changing sequences.

Another aspect that has been discussed, but not experimentally investigated, is the use of *traffic-actuated controls* within the SuOC. As the observer/controller requires detection capabilities anyway, it would be a logical consequence to utilise the detection data for traffic-actuated control. By adapting the temporal and logical conditions that define the signalisation, the observer/controller provides the missing on-line learning and optimisation capabilities. Although the handling of traffic-actuated controls in the SuOC does not present a conceptual challenge, technical

hurdles caused by deficiencies of the simulator's programming interface hindered an experimental evaluation within this thesis.

9.2.2 Observer/controller architecture

Regarding the generic observer/controller architecture, promising directions for future research include a faster on-line learning, the improved support of changing system objectives, and the incorporation of automated calibration techniques for the controller's simulation model.

Learning speed

Important characteristics of the observer/controller architecture are its learning and optimisation capabilities. Although the generic framework does not explicitly specify suitable mechanisms, LCSs have been widely used to implement the controller's on-line learning capabilities (see, e. g., [159, 193]). As rule-based reinforcement learners, LCSs are well-suited to provide a situation-action mapping, but they are also known to require plenty of learning interactions. This poses a serious difficulty for applications where reinforcement is scarce.

In this thesis, scarce reinforcement signals have been addressed by two-levelled learning. Other approaches improve the learning speed of LCSs by distribution [10, 12, 53, 82, 158] or deduction [72]. *Distributed classifier systems* divide a given problem into subproblems, assign each subproblem to an LCS, and combine the individual results to reach a decision for the original problem. *Deduction mechanisms* infer general, but accurate classifiers by intelligently combining rules from the existing population.

As neither distribution nor deduction avoid a negative influence of learning errors on the controlled system, both approaches cannot replace two-levelled learning. Nevertheless, distributed or deductive classifier systems can speed up the controller's on-line learning whenever the application allows for a problem decomposition or can be handled by binary classifiers. First investigations in this direction have been presented in [72, 158], but the transfer to a complex real-world problem (like traffic signal control) and the combination with a simulation-based optimisation heuristic remain challenging open issues.

However, the refinement of LCSs is not the only possible way to speed up two-levelled learning. Other learning and optimisation techniques (like *Gaussian processes* [151] for on-line learning or *Particle Swarm Optimisation* [101] for off-line optimisation) constitute building blocks that can be combined in future versions of the two-levelled mechanism.

Flexibility

An issue that has not been explicitly addressed in previous observer/controller research is flexibility with respect to changing objectives: An observer/controller acts according to a user-defined objective function. When the objective changes, the change needs to be reflected in the controller's situation-action mapping.

A simple way to support changing objectives is to keep track of a separate situation-action mapping for every objective and switch among them. However, only the active mapping is improved by two-levelled learning such that the quality of the mappings differs. A mapping for an objective that is pursued only infrequently will thus be of a relatively low quality. Rarely used objectives are, however, often related to exceptional situations where a good performance is of special importance. Considering signalised intersections as example, throughput maximisation is important in case of a high demand, while the goal is to minimise delays in the majority of situations.

Due to the importance of flexibility in many OC scenarios, the development of an observer/controller implementation with an improved support of changing objectives is an interesting direction for future research.

Model calibration

One of the outstanding features of the observer/controller framework is the combination of on-line learning and model-based off-line optimisation. This two-levelled mechanism works best when the output of the model closely resembles the observations in the SuOC. In this thesis, the controller's LCS detects and corrects potential discrepancies based on reinforcement observed in the SuOC. Unfortunately, the correction requires the activation of incorrectly evaluated actions in the SuOC and does not

correct the causative model error. By integrating an automated calibration mechanism into the observer/controller, the model error could be addressed directly. The on-line calibration of simulation models has been successfully addressed in varying application areas including robotics [20] and microscopic traffic simulation [124]. The transfer of these results into an observer/controller could improve the framework.

9.2.3 Self-organised collaboration

Further potential for improvement lies in the collaboration of distributed observer/controller components. In the context of traffic signal control, collaborative learning, optimal coordination, and dynamic route guidance are among the most promising research directions.

Collaborative learning

When distributed observer/controller components have to learn similar tasks, collaboration is a promising approach to speed up the learning process. Collaborative learning can be implemented with the help of a public situation-action mapping or by the exchange of learnt rules.

A *public mapping* is shared and commonly updated by several observer/controllers [73]. Compared to a private mapping maintained by a single learner, the public mapping receives a much higher number of updates in the same time span. Thus, learning can be accelerated whenever several observer/controller components are occupied with the same task.

Rule exchange [32] constitutes a second approach for collaborative learning. Here, the observer/controller components maintain their individual situation-action mappings, but exchange selected mapping entries to benefit from each others' experience. In contrast to the use of a public mapping, rule exchange can to some extent consider individual properties of the collaborating learners.

The integration of collaborative learning techniques into a group of distributed observer/controller components is an interesting field of research. In a road traffic network, several organic intersections could utilise the existing communication infrastructure to exchange learnt signal plans. The heterogeneity of signalised intersections will be a challenge, though.

Optimal signal coordination

The hierarchical coordination of signalised intersections proposed in this thesis is based on a heuristic approach. Although experiments show that the mechanism works well, it cannot guarantee an optimal coordination. As the mathematically exact optimisation of offsets is possible [108, 109], it would be interesting to replace the RM's heuristic. In contrast to the heuristic that merely identifies a set of non-conflicting traffic streams, the offset optimisation algorithm provides a set of optimal offset for the network's intersections. However, it requires a predefined signal plan for every intersection as input. Furthermore, the signal plans need to have a common cycle length. Although a combination of local signal plan selection and decentralised cycle time negotiation with a regional offset optimisation is promising, extensive modifications to the current hierarchical system will be required.

Dynamic route guidance

Another way to benefit from collaborating intersections in a road network is to setup a system for dynamic route guidance. Similar to SCOOT [98] or MOTION [112], organic intersections can combine their data on traffic flows and green times to support the traffic-responsive routing of vehicles through the network. Utilising the communication infrastructure assumed for the DPSS mechanism, internet routing protocols like *Distance Vector Routing* or *Link State Routing* (see, e. g., [187]) can be adapted to implement a self-organised routing mechanism. Thereby, organic intersections are no longer limited to react on observed traffic flows, but can additionally influence the route choice of the drivers. First steps towards dynamic route guidance have been presented in [148], but further investigations with respect to an improved incident detection are required.

9.3 Final remarks

Despite the numerous questions remaining for further research, this thesis has made first steps towards a network of autonomous, self-organising intersections. Its major contribution is a successful application of the

observer/controller architecture in the context of a signalised road network. With the help of the observer/controller, the optimisation of signal plans has been shifted from the design time to the run-time of a signal system, while a self-organising collaboration mechanism keeps track of the network's signal coordination. As a result, vehicular delays and stops in a network can be reduced while the robustness of the signal system with respect to changing demands is improved.

The observer/controller's core feature is a newly developed two-levelled learning mechanism that combines on-line reinforcement learning and model-based off-line optimisation. In this thesis, the mechanism has been successfully applied to a complex and highly relevant real-world problem. However, two-levelled learning is not restricted to traffic control, but can be applied to numerous safety- and performance critical environments. As many of these environments have been inept for on-line optimisation and learning before, two-levelled learning opens new application domains for machine learning and Organic Computing.

References

[1] G. Abu-Lebdeh. Genetic algorithms. In *Artificial Intelligence in Transportation*, Transportation Research Circular E-C113, pages 49–71. Transportation Research Board, 2007.

[2] G. Abu-Lebdeh and B. H. Al-Omari. Configuring micro-genetic algorithms for solving traffic control problems: The case of number of generations. In *Proceedings of the 4th International Symposium on Uncertainty Modeling and Analysis (ISUMA'03)*, pages 70–77. IEEE, 2003.

[3] G. Abu-Lebdeh and R. F. Benekohal. Convergence variability and population sizing in micro-genetic algorithms. *Computer-Aided Civil and Infrastructure Engineering*, 14(5):321–334, 1999.

[4] R. Akçelik. Traffic signals: Capacity and timing analysis. Technical Report 123, Australian Road Research Board, 1981.

[5] E. Alba and M. Tomassini. Parallelism and evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5):443–460, 2002.

[6] E. Almasri. *A new offset optimization method for signalized urban road networks.* Dissertation, Fakultät für Bauingenieurwesen und Geodäsie, Universität Hannover, 2006.

[7] E. Almasri and B. Friedrich. Online offset optimisation in urban networks based on cell transmission model. In *Proceedings of the 5th European Congress and Exhibition on Intelligent Transport Systems and Services (ITS05).* ERTICO – ITS Europe, 2005.

[8] E. Avineri. Soft computing applications in traffic and transport systems: A review. In F. Hoffmann, M. Köppen, F. Klawonn, and R. Roy, editors, *Soft Computing: Methodologies and Applications*, volume 32 of *Advances in Soft Computing*, pages 17–25. Springer, 2005.

[9] T. Bäck, D. B. Fogel, and Z. Michalewicz, editors. *Evolutionary Computation 1: Basic Algorithms and Operators.* Institute of Physics Publishing, 2000.

[10] S. M. Baneamoon, R. A. Salam, and A. Z. H. Talib. Learning process enhancement for robot behaviors. *International Journal of Intelligent Technology*, 2(3):172–177, 2007.

[11] J. Barceló, E. Codina, J. Casas, J. Ferrer, and D. García. Microscopic traffic simulation: A tool for the design, analysis and evaluation of intelligent transport systems. *Journal of Intelligent and Robotic Systems*, 41(2–3):173–203, 2005.

[12] A. Barry. Hierarchy formation within classifier systems – A review. In *Proceedings of the 1st International Conference on Evolutionary Algorithms and their Applications (EVCA 1996)*, pages 195–211, 1996.

[13] A. L. C. Bazzan. A distributed approach for coordination of traffic signal agents. *Autonomous Agents and Multi-Agent Systems*, 10(2):131–164, 2005.

[14] A. L. C. Bazzan. Opportunities for multiagent systems and multiagent reinforcement learning in traffic control. *Autonomous Agents and Multi-Agent Systems*, 18(3):342–375, 2009.

[15] A. L. C. Bazzan and F. Klügl, editors. *Multi-Agent Systems for Traffic and Transportation Engineering*. Information Science Reference, 2009.

[16] D. M. Beazley. *Python Essential Reference*. Addison Wesley, 4th edition, 2009.

[17] M. C. Bell and R. D. Bretherton. Ageing of fixed-time traffic signal plans. In *Proceedings of the 2nd IEE Conference on Road Traffic Control*, pages 77–80, 1986.

[18] R. E. Bellman and L. A. Zadeh. Decision-making in a fuzzy environment. *Management Science*, 17(4):141–164, 1970.

[19] L. Bloomberg, M. Swenson, and B. Haldors. Comparison of simulation models and the HCM. In *Proceedings of the 82nd Transportation Research Board Annual Meeting*. Transportation Research Board, 2003.

[20] J. Bongard, V. Zykov, and H. Lipson. Resilient machines through continuous self-modeling. *Science*, 314(5802):1118–1121, 2006.

[21] J. Branke. *Evolutionary Optimization in Dynamic Environments*, volume 3 of *Genetic Algorithms and Evolutionary Computation*. Kluwer Academic Publishers, 2002.

[22] J. Branke, P. Goldate, and H. Prothmann. Actuated traffic signal optimization using evolutionary algorithms. In *Proceedings of the 6th European Congress and Exhibition on Intelligent Transport Systems and Services (ITS07)*. ERTICO – ITS Europe, 2007.

[23] J. Branke, M. Mnif, C. Müller-Schloer, H. Prothmann, U. Richter, F. Rochner, and H. Schmeck. Organic Computing – Addressing complexity by controlled self-organization. In T. Margaria, A. Philippou, and B. Steffen, editors, *Post-Conference Proceedings of the 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2006)*, pages 185–191. IEEE, 2006.

[24] U. Brannolte. Simulationsmodelle im Verkehrswesen – Analyse und spezifische Weiterentwicklungen. *Straßenverkehrstechnik*, 44(6):265–269, 2000.

[25] R. Braun. *Ein echtzeitfähiger Evolutionärer Algorithmus zur netzweiten Optimierung der Lichtsignalsteuerung.* Dissertation, Lehrstuhl für Verkehrstechnik, Technische Universität München, 2008.

[26] R. Braun, C. Kemper, and F. Weichenmeier. TRAVOLUTION – Adaptive urban traffic signal control with an evolutionary algorithm. In *Proceedings of the 4th International Symposium "Networks for Mobility"*, 2008.

[27] R. Braun, S. Männicke, and F. Weichenmeier. Anwendung und Bewertung eines Verfahrens zur netzweiten Offline-Optimierung der Lichtsignal-Koordinierung mittels genetischer Algorithmen. In *Tagungsband Verkehrswissenschaftliche Tage*, 2005.

[28] R. Braun and F. Weichenmeier. Automatic offline-optimization of coordinated traffic signal control in urban networks using genetic algorithms. In *Proceedings of the 12th World Congress on Intelligent Transport Systems*, 2005.

[29] R. Braun and F. Weichenmeier. Automatische Offline-Optimierung der lichtsignaltechnischen Koordinierung des mIV im städtischen Netz unter Verwendung genetischer Algorithmen. *Straßenverkehrstechnik*, 49(5):232–238, 2005.

[30] R. D. Bretherton and G. I. Rai. The use of SCOOT in low flow conditions. *Traffic engineering and control*, 23(12):574–576, 1982.

[31] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, 1986.

[32] O. Buchtala and B. Sick. Functional knowledge exchange within an intelligent distributed system. In P. Lukowicz, L. Thiele, and G. Tröster, editors, *Architecture of Computing Systems (ARCS 2007) – 20th International Conference*, volume 4415 of *LNCS*, pages 126–141. Springer, 2007.

[33] L. Bull, editor. *Applications of Learning Classifier Systems*, volume 150 of *Studies in Fuzziness and Soft Computing*. Springer, 2004.

[34] L. Bull, J. Sha'Aban, A. Tomlinson, J. D. Addison, and B. Heydecker. Towards distributed adaptive control for road traffic junction signals using learning classifier systems. In L. Bull, editor, *Applications of Learning Classifier Systems*, pages 276–299. Springer, 2004.

[35] H.-J. Bungartz, S. Zimmer, M. Buchholz, and D. Pflüger. *Modellbildung und Simulation – Eine anwendungsorientierte Einführung*. Springer, 2009.

[36] S. Burmester, H. Giese, E. Münch, O. Oberschelp, F. Klein, and P. Scheideler. Tool support for the design of self-optimizing mechatronic multi-agent systems. *International Journal on Software Tools for Technology Transfer*, 10(3):207–222, 2008.

[37] M. V. Butz. *Rule-Based Evolutionary Online Learning Systems – A Principled Approach to LCS Analysis and Design*, volume 191 of *Studies in Fuzziness and Soft Computing*. Springer, 2005.

[38] M. V. Butz, D. E. Goldberg, and K. Tharakunnel. Analysis and improvement of fitness exploitation in XCS: Bounding models, tournament selection, and bilateral accuracy. *Evolutionary Computation*, 11(3):239–277, 2003.

[39] M. V. Butz, T. Kovacs, P. L. Lanzi, and S. W. Wilson. How XCS evolves accurate classifiers. In L. Spector et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '01)*, pages 927–934, 2001.

[40] M. V. Butz and S. W. Wilson. An algorithmic description of XCS. *Soft Computing*, 6(3–4):144–153, 2002.

[41] E. F. Camacho and C. Bordons. *Model Predictive Control*. Springer, 2nd edition, 2004.

[42] E. Cantú-Paz. *Efficient and Accurate Parallel Genetic Algorithms*, volume 1 of *Genetic Algorithms and Evolutionary Computation*. Springer, 2000.

[43] Y. J. Cao, N. Ireson, L. Bull, and R. Miles. Design of a traffic junction controller using classifier system and fuzzy logic. In B. Reusch, editor, *Computational Intelligence – Theory and Applications*, volume 1625 of *LNCS*, pages 342–353. Springer, 1999.

[44] Y. J. Cao, N. Ireson, L. Bull, and R. Miles. Distributed learning control of traffic signals. In S. Cagnoni et al., editors, *Real-World Applications of Evolutionary Computing – EvoWorkshops Proceedings*, volume 1803 of *LNCS*, pages 117–126. Springer, 2000.

[45] J. Casas, J. L. Ferrer, D. Garcia, J. Perarnau, and A. Torday. Traffic Simulation with Aimsun. In J. Barceló, editor, *Fundamentals of Traffic Simulation*, volume 145 of *International Series in Operations Research & Management Science*, pages 173–232. Springer, 2010.

[46] E. J. H. Chang. Echo algorithms: Depth parallel operations on general graphs. *IEEE Transactions on Software Engineering*, 8(4):391–401, 1982.

[47] R. Chrobok, O. Kaumann, J. Wahle, and M. Schreckenberg. Different methods of traffic forecast based on real data. *European Journal of Operational Research*, 155(3):558–568, 2004.

[48] S.-B. Cools, C. Gershenson, and B. D'Hooghe. Self-organizing traffic lights: A realistic simulation. In M. Prokopenko, editor, *Advances in Applied Self-organizing Systems*, pages 41–50. Springer, 2007.

[49] B. C. da Silva, D. de Oliveira, A. L. C. Bazzan, and E. W. Basso. Adaptive traffic control with reinforcement learning. In *Proceedings of the 4th Workshop on Agents in Traffic and Transportation (ATT 06)*, pages 80–86, 2006.

[50] C. F. Daganzo. The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory.

Transportation Research Part B: Methodological, 28(4):269–287, 1994.

[51] C. F. Daganzo. The cell transmission model, Part II: Network traffic. *Transportation Research Part B: Methodological*, 29(2):79–93, 1995.

[52] H. H. Dam, H. A. Abbass, and C. Lokan. Be real! XCS with continuous-valued inputs. In F. Rothlauf et al., editors, *Proceedings of the 2005 Workshops on Genetic and Evolutionary Computation (GECCO '05)*, pages 85–87. ACM, 2005.

[53] H. H. Dam, H. A. Abbass, and C. Lokan. DXCS: An XCS system for distributed data mining. In H.-G. Beyer et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '05)*, pages 1883–1890. ACM, 2005.

[54] K. De Jong. Learning with genetic algorithms: An overview. *Machine Learning*, 3(2–3):121–138, 1988.

[55] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley, 2001.

[56] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[57] C. Diakaki, V. Dinopoulou, K. Aboudolas, M. Papageorgiou, E. Ben-Shabat, E. Seider, and A. Leibov. Extensions and new applications of the traffic signal control strategy TUC. In *Transportation Research Record 1856*, pages 202–211. Transportation Research Board, 2003.

[58] F. Dion, H. Rakha, and Y.-S. Kang. Comparison of delay estimates at under-saturated and over-saturated pre-timed signalized intersections. *Transportation Research Part B: Methodological*, 38(2):99–122, 2004.

[59] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, 2004.

[60] R. Dowling, A. Skabardonis, and V. Alexiadis. *Traffic Analysis Toolbox Volume III: Guidelines for Applying Traffic Microsimulation Modeling Software*. Federal Highway Administration, 2004.

[61] K. Dresner and P. Stone. A multiagent approach to autonomous intersection management. *Journal of Artificial Intelligence Research*, 31(1):591–656, 2008.

[62] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer, 2nd edition, 2007.

[63] W. Elmenreich and H. de Meer. Self-organizing networked systems for technical applications: A discussion on open issues. In K. A. Hummel and J. P. G. Sterbenz, editors, *Self-Organizing Systems – Third International Workshop, IWSOS 2008*, volume 5343 of *LNCS*, pages 1–9. Springer, 2008.

[64] G. Enee and C. Escazut. Classifier systems evolving multi-agent system with distributed elitism. In *Proceedings of the 1999 Congress on Evolutionary Computation (CEC 99)*, pages 1740–1746. IEEE, 1999.

[65] J. Esser and M. Schreckenberg. Microscopic simulation of urban traffic based on cellular automata. *International Journal of Modern Physics C*, 8(5):1025–1036, 1997.

[66] M. Fellendorf and P. Vortisch. Microscopic Traffic Flow Simulator VISSIM. In J. Barceló, editor, *Fundamentals of Traffic Simulation*, volume 145 of *International Series in Operations Research & Management Science*, pages 63–93. Springer, 2010.

[67] L. J. A. Ferrerira. Car fuel consumption in urban traffic: The results of a survey in Leeds using instrumented vehicles. ITS Working Paper 162, Institute for Transportation Studies, University of Leeds, 1982.

[68] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. Wiley, 1966.

[69] Forschungsgesellschaft für Straßen- und Verkehrswesen. Handbuch für die Bemessung von Straßenverkehrsanlagen (HBS) – Ausgabe 2001, Fassung 2009. FGSV Verlag, 2009.

[70] Forschungsgesellschaft für Straßen- und Verkehrswesen. Richtlinien für Lichtsignalanlagen (RiLSA) – Ausgabe 2010. FGSV Verlag, 2010.

[71] M. D. Foy, R. F. Benekohal, and D. E. Goldberg. Signal timing determination using genetic algorithms. In *Transportation Research Record 1365*, pages 108–115. Transportation Research Board, 1992.

[72] N. Fredivianus, H. Prothmann, and H. Schmeck. XCS revisited: A novel discovery component for the eXtended Classifier System. In K. Deb et al., editors, *Simulated Evolution and Learning – 8th International Conference (SEAL 2010)*, volume 6457 of *LNCS*, pages 289–298. Springer, 2010.

[73] N. Fredivianus, U. Richter, and H. Schmeck. Collaborating and learning predators on a pursuit scenario. In M. Hinchey, B. Kleinjohann, L. Kleinjohann, P. Lindsay, F. Rammig, J. Timmis, and M. Wolf, editors, *Distributed, Parallel and Biologically Inspired Systems*, volume 329 of *IFIP Advances in Information and Communication Technology*, pages 290–301. Springer, 2010.

[74] B. Friedrich. Steuerung von Lichtsignalanlagen: BALANCE – ein neuer Ansatz. *Straßenverkehrstechnik*, 44(7):321–328, 2000.

[75] B. Friedrich. Verkehrsadaptive Steuerung von Lichtsignalanlagen – Ein Überblick. In *Festschrift zum Ehrenkolloquium für Univ.-Prof. Dr./UCB Hartmut Keller*. Fachgebiet Verkehrstechnik und Verkehrsplanung der Technischen Universität München, 2002.

[76] B. Friedrich and E. Almasri. Modellbasierte Optimierung der Versatzzeiten mit dem Cell Transmission Model. In *HEUREKA '05, Optimierung in Verkehr und Transport – Tagungsband*, pages 331–346. FGSV Verlag, 2005.

[77] B. Friedrich and E. Almasri. Modellbasierte Optimierung der Versatzzeiten mit dem Cell Transmission Model. *Straßenverkehrstechnik*, 49(4):169–175, 2005.

[78] N. H. Gartner. OPAC Strategy for demand-responsive decentralized traffic signal control. In J.-P. Perrin, editor, *Control, Computers, Communications in Transportation*, pages 241–244, 1989.

[79] N. H. Gartner, L. Zhang, and H. Li. Comparative evaluation of three adaptive control strategies: OPAC, TACOS, and FLC. In *Proceedings of the 85th Transportation Research Board Annual Meeting*. Transportation Research Board, 2006.

[80] C. Gershenson. Self-organizing traffic lights. *Complex Systems*, 16(1):29–53, 2005.

[81] C. Gershenson. *Design and Control of Self-organizing Systems*. PhD thesis, Faculteit Wetenschappen, Vrije Universiteit Brussel, 2007.

[82] M. Gershoff and S. Schulenburg. Collective behavior based hierarchical XCS. In *Proceedings of the 2007 Genetic And Evolutionary Computation Conference (GECCO '07)*, pages 2695–2700. ACM, 2007.

[83] P. G. Gipps. A behavioural car-following model for computer simulation. *Transportation Research Part B: Methodological*, 15(2):105–111, 1981.

[84] P. G. Gipps. A model for the structure of lane-changing decisions. *Transportation Research Part B: Methodological*, 20(5):403–414, 1986.

[85] P. G. Gipps. MULTSIM: A model for simulate vehicular traffic on multi-lane arterial roads. *Mathematics and computers in simulation*, 28(4):291–295, 1986.

[86] M. Girianna and R. F. Benekohal. Dynamic signal coordination for networks with oversaturated intersections. In *Transportation*

Research Record 1811, pages 122–130. Transportation Research Board, 2002.

[87] M. Girianna and R. F. Benekohal. Intelligent signal coordination on congested networks using parallel micro genetic algorithms. In K. C. P. Wang, S. Madanat, S. Nambisan, and G. Spring, editors, *Proceedings of the 7th International Conference on Applications of Advanced Technologies in Transportation*, 2002.

[88] M. Girianna and R. F. Benekohal. Signal coordination for a two-way street network with oversaturated intersections. In *Proceedings of the 82nd Transportation Research Board Annual Meeting*. Transportation Research Board, 2003.

[89] M. Girianna and R. F. Benekohal. Solving signal coordination problems using master-slave genetic algorithms. In *Proceedings of the 82nd Transportation Research Board Annual Meeting*. Transportation Research Board, 2003.

[90] M. Girianna and R. F. Benekohal. Using genetic algorithms to design signal coordination for oversaturated networks. *Journal of Intelligent Transportation Systems*, 8(2):117–129, 2004.

[91] J. Gosling, B. Joy, G. Steele, and G. Bracha. *The Java Language Specification*. Addison Wesley, 3rd edition, 2005.

[92] J. J. Grefenstette and C. L. Ramsey. An approach to anytime learning. In *Proceedings of the 9th International Workshop on Machine Learning*, pages 189–195, 1992.

[93] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 2nd edition, 1998.

[94] F. Höfler. *Verkehrswesen-Praxis*. Bauwerk, 2006.

[95] J. H. Holland. Adaptation. In R. Rosen and F. Snell, editors, *Progress in Theoretical Biology IV*, pages 263–293. Academic Press, 1976.

[96] J. H. Holland. *Adaptation in Natural and Artificial Systems.* MIT Press, 2nd edition, 1992.

[97] J. H. Holland and J. S. Reitman. Cognitive systems based on adaptive algorithms. In D. A. Waterman and F. Hayes-Roth, editors, *Pattern directed inference systems*, pages 313–329. Academic Press, 1978.

[98] N. B. Hounsell, M. McDonald, and R. A. Lambert. The integration of SCOOT and dynamic route guidance. In *Proceedings of the IEEE Road Traffic Monitoring Conference*, pages 168–172, 1992.

[99] J. Hourdakis and P. G. Michalopoulos. Evaluation of ramp control effectiveness in two twin cities freeways. In *Transporation Research Record 1811*, pages 21–29. Transportation Research Board, 2002.

[100] V. F. Hurdle. Signalized intersection delay models: A primer for the uninitiated. In *Transportation Research Record No. 971*, pages 96–105. Transportation Research Board, 1984.

[101] J. Kennedy and R. C. Eberhart. *Swarm Intelligence.* Morgan Kaufmann, 2001.

[102] J. O. Kephart and D. M. Chess. The vision of Autonomic Computing. *IEEE Computer*, 36(1):41–50, 2003.

[103] K. B. Kesur. Advances in genetic algorithm optimization of traffic signals. Master's thesis, Faculty of Science, University of the Witwatersrand, 2007.

[104] K. B. Kesur. Advances in genetic algorithm optimization of traffic signals. *Journal of Transportation Engineering*, 135(4):160–173, 2009.

[105] K.-O. Kim and L. R. Rilett. Simplex-based calibration of traffic microsimulation models with intelligent transportation systems data. In *Transportation Research Record 1855*, pages 80–89. Transportation Research Board, 2003.

[106] F. Klügl, A. Bazzan, and S. Ossowski, editors. *Applications of Agent Technology in Traffic and Transportation.* Birkhäuser, 2005.

[107] J. Knowles. ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1):50–66, 2006.

[108] E. Köhler, R. H. Möhring, K. Nökel, and G. Wünsch. Optimization of signalized traffic networks. In W. Jäger and H.-J. Krebs, editors, *Mathematics – Key Technology for the Future*, pages 179–188. Springer, 2008.

[109] E. Köhler, R. H. Möhring, and G. Wünsch. Minimizing total delay in fixed-time controlled traffic networks. In H. Fleuren, D. den Hertog, and P. Kort, editors, *Proceedings of Operations Research (OR) 2004*, pages 192–199, 2004.

[110] J. R. Koza. *Genetic Programming.* MIT Press, 1992.

[111] J. R. Koza. *Genetic Programming II.* MIT Press, 1994.

[112] G. Kruse. COSMOS – Results of the MOTION demonstrator for congestion and incident management strategies in Piraeus. In *Proceedings of Trafikdage 1999*, 1999.

[113] G. Kruse. MOTION Netzsteuerung – Optimierung der Lichtsignalsteuerung im Einsatz. In M. Steierwald and S. Martens, editors, *Steuerung kommunaler Verkehrsnetze*, pages 37–52. Akademie für Technikfolgenabschätzung in Baden-Württemberg, 2003.

[114] S. Lämmer. *Reglerentwurf zur dezentralen Online-Steuerung von Lichtsignalanlagen in Straßennetzwerken.* Dissertation, Fakultät Verkehrswissenschaften „Friedrich List“, Technische Universität Dresden, 2007.

[115] S. Lämmer. Stabilitätsprobleme voll-verkehrsabhängiger Lichtsignalsteuerungen. Technical report, Technische Universität Dresden, 2009.

[116] S. Lämmer and D. Helbing. Self-control of traffic lights and vehicle flows in urban road networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(4):P04019, 2008.

[117] S. Lämmer, J. Krimmling, and A. Hoppe. Lichtsignalanlagen: Selbst-Steuerung von Lichtsignalanlagen in Straßennetzwerken – Regelungstechnischer Ansatz und Simulation. *Straßenverkehrstechnik*, 53(11):714–721, 2009.

[118] P. L. Lanzi. Learning classifier systems: then and now. *Evolutionary Intelligence*, 1(1):63–82, 2008.

[119] P. L. Lanzi, D. Loiacono, S. W. Wilson, and D. E. Goldberg. Generalization in the XCSF classifier system: Analysis, improvement, and extension. *Evolutionary Computation*, 15(2):133–168, 2007.

[120] S. Liang. *The Java Native Interface – Programmer's Guide and Specification*. Addison Wesley, 1999.

[121] M. J. Lighthill and G. B. Whitham. On kinematic waves II. A theory of traffic flow on long crowded roads. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 229(1178):317–345, 1955.

[122] L. Liu, S. Thanheiser, and H. Schmeck. A reference architecture for self-organizing service-oriented computing. In U. Brinkschulte, T. Ungerer, C. Hochberger, and R. G. Spallek, editors, *Architecture of Computing Systems (ARCS 2008) – 21st International Conference*, volume 4934 of *LNCS*, pages 205–219. Springer, 2008.

[123] Z. Liu. A survey of intelligence methods in urban traffic signal control. *International Journal of Computer Science and Network Security*, 7(7):105–112, 2007.

[124] S. Lorkowski and P. Wagner. Parameter calibration of traffic models in microscopic on-line simulations. In *Proceedings of the 84th Transportation Research Board Annual Meeting*. Transportation Research Board, 2005.

[125] P. R. Lowrie. The Sydney Co-ordinated Adaptive Traffic System – Principles, methodology, algorithms. In *Proceedings of the International Conference on Road Traffic Signalling*, pages 67–70, 1982.

[126] M. Mazzamatti, D. V. V. F. Netto, L. Vilanova, and S. Ming. Benefits gained by responsive and traffic adaptive systems in São Paulo. In *Proceedings of the 9th International Conference on Road Transport Information and Control*, pages 114–118. IEEE, 1998.

[127] J. Mertz. *Ein mikroskopisches Verfahren zur verkehrsadaptiven Knotenpunktsteuerung mit Vorrang des öffentlichen Verkehrs*. Dissertation, Fachgebiet Verkehrstechnik und Verkehrsplanung, Technische Universität München, 2001.

[128] Z. Michalewicz. *Genetic algorithms + Data structures = Evolution programs*. Springer, 3rd edition, 1999.

[129] D. L. Mills. Network time protocol (version 3) – Specification, implementation and analysis. Technical Report 90-6-1, Electrical Engineering Department, University of Delaware, 1990.

[130] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[131] M. Mnif, U. Richter, J. Branke, H. Schmeck, and C. Müller-Schloer. Measurement and control of self-organised behaviour in robot swarms. In P. Lukowicz, L. Thiele, and G. Tröster, editors, *Architecture of Computing Systems (ARCS 2007) – 20th International Conference*, volume 4415 of *LNCS*, pages 209–223. Springer, 2007.

[132] L. Montero, E. Codina, J. Barceló, and P. Barceló. A combined methodology for transportation planning assessment. Application to a case study. *Transportation Research Part C: Emerging Technologies*, 9(3):213–230, 2001.

[133] D. C. Montgomery. *Design and Analysis of Experiments*. Wiley, 7th edition, 2008.

[134] J. Mück. BALANCE: Adaptive Lichtsignalsteuerungen für Straßennetze und einzelne Knotenpunkte. In M. Steierwald and

S. Martens, editors, *Steuerung kommunaler Verkehrsnetze*, pages 53–66. Akademie für Technikfolgenabschätzung in Baden-Württemberg, 2003.

[135] J. Mück. Verkehrssteuerung – Neue Schätz- und Optimierungsverfahren für Adaptive Netzsteuerungen. *Straßenverkehrstechnik*, 52(12):761–773, 2008.

[136] C. Müller-Schloer. Organic Computing: On the feasibility of controlled emergence. In *Proceedings of the 2nd International Conference on Hardware/Software Codesign and System Synthesis*, pages 2–5, 2004.

[137] C. Müller-Schloer and H. Schmeck. Organic Computing: A grand challenge for mastering complex systems. *it – Information Technology*, 52(3):135–141, 2010.

[138] C. Müller-Schloer, H. Schmeck, and T. Ungerer, editors. *Organic Computing – A Paradigm Shift for Complex Systems.* Autonomic Systems. Birkhäuser, 2011.

[139] C. Müller-Schloer, C. von der Malsburg, and R. P. Würtz. Organic Computing. *Informatik Spektrum*, 27(4):332–336, 2004.

[140] K. Nagel and M. Schreckenberg. A cellular automaton model for freeway traffic. *Journal de Physique*, 2(12):2221–2229, 1992.

[141] National Electrical Manufacturers Association. NEMA Standards Publication TS 2-2003 v02.06 – Traffic Controller Assemblies with NTCIP Requirements, 2003.

[142] O. Oberschelp, T. Hestermeyer, B. Kleinjohann, and L. Kleinjohann. Design of self-optimizing agent-based controllers. In *Proceedings of the Workshop 2002 – Agent-Based Simulation 3*, 2002.

[143] K. Ohno and H. Mine. Optimal traffic signal settings – II. A refinement of Webster's method. *Transportation Research*, 7(3):269–292, 1973.

[144] M. Papageorgiou, C. Diakaki, V. Dinopoulou, A. Kotsialos, and Y. Wang. Review of road traffic control strategies. *Proceedings of the IEEE*, 91(12):2043–2067, 2003.

[145] H. Prothmann, J. Branke, H. Schmeck, S. Tomforde, F. Rochner, J. Hähner, and C. Müller-Schloer. Organic traffic light control for urban road networks. *International Journal of Autonomous and Adaptive Communications Systems*, 2(3):203–225, 2009.

[146] H. Prothmann, F. Rochner, S. Tomforde, J. Branke, C. Müller-Schloer, and H. Schmeck. Organic control of traffic lights. In C. Rong, M. G. Jaatun, F. E. Sandnes, L. T. Yang, and J. Ma, editors, *Autonomic and Trusted Computing – 5th International Conference (ATC 2008)*, volume 5060 of *LNCS*, pages 219–233. Springer, 2008.

[147] H. Prothmann and H. Schmeck. Evolutionary algorithms for traffic signal optimisation: A survey. In *Proceedings of mobil.TUM 2009 – International Scientific Conference on Mobility and Transport (CD-ROM)*, 2009.

[148] H. Prothmann, S. Tomforde, J. Branke, J. Hähner, C. Müller-Schloer, and H. Schmeck. Organic traffic control. In C. Müller-Schloer, H. Schmeck, and T. Ungerer, editors, *Organic Computing – A Paradigm Shift for Complex Systems*, Autonomic Systems, chapter 5.1, pages 431–446. Birkhäuser, 2011.

[149] QUARTET Deliverable No. 2. Assessment of current tools for environmental assessment in QUARTET, DRIVE II Project V2018: QUARTET, 1992.

[150] C. L. Ramsey and J. J. Grefenstette. Case-based anytime learning. In *Case-Based Reasoning: Papers from the 1994 Workshop*, 1994.

[151] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[152] A. K. Rathi. The use of common random numbers to reduce the variance in network simulation of traffic. *Transportation Research Part B: Methodological*, 26(5):357–363, 1992.

[153] A. K. Rathi and A. J. Santiago. Urban network traffic simulations: TRAF-NETSIM program. *Journal of Transportation Engineering*, 116(6):734–743, 1990.

[154] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Frommann-Holzboog, 1973.

[155] O. Ribock, U. Richter, and H. Schmeck. Using Organic Computing to control bunching effects. In U. Brinkschulte, T. Ungerer, C. Hochberger, and R. G. Spallek, editors, *Architecture of Computing Systems (ARCS 2008) – 21st International Conference*, volume 4934 of *LNCS*, pages 232–244. Springer, 2008.

[156] P. I. Richards. Shock waves on the highway. *Operations Research*, 4(1):42–51, 1956.

[157] U. Richter, M. Mnif, J. Branke, C. Müller-Schloer, and H. Schmeck. Towards a generic observer/controller architecture for organic computing. In C. Hochberger and R. Liskowsky, editors, *Informatik 2006 – Informatik für Menschen*, volume P-93 of *LNI*, pages 112–119. Köllen Verlag, 2006.

[158] U. Richter, H. Prothmann, and H. Schmeck. Improving XCS performance by distribution. In X. Li, M. Zhang, and M. Kirley, editors, *Simulated Evolution and Learning – 7th International Conference (SEAL 2008)*, volume 5361 of *LNCS*, pages 111–120. Springer, 2008.

[159] U. M. Richter. *Controlled Self-organisation using Learning Classifier Systems.* KIT Scientific Publishing, 2009.

[160] D. I. Robertson and R. D. Bretherton. Optimizing networks of traffic signals in real time – the SCOOT method. *IEEE Transactions on Vehicular Technology*, 40(1):11–15, 1991.

[161] F. Rochner, H. Prothmann, J. Branke, C. Müller-Schloer, and H. Schmeck. An organic architecture for traffic light controllers. In C. Hochberger and R. Liskowsky, editors, *Informatik 2006 – Informatik für Menschen*, volume P-93 of *LNI*, pages 120–127. Köllen Verlag, 2006.

[162] J. Sánchez, M. Galán, and E. Rubio. Genetic algorithms and cellular automata: A new architecture for traffic light cycles optimization. In *Proceedings of the 2004 Congress on Evolutionary Computation (CEC 04)*, pages 1668–1674. IEEE, 2004.

[163] J. Sánchez, M. Galán, and E. Rubio. Bit level versus gene level crossover in a traffic modeling environment. In *Proceedings of the 2005 International Conference on Computational Intelligence for Modelling, Control and Automation, and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'05)*, pages 1190–1195. IEEE, 2005.

[164] J. Sánchez, M. Galán, and E. Rubio. Stochastic vs deterministic traffic simulator. Comparative study for its use within a traffic light cycles optimization architecture. In J. Mira and J. R. Álvarez, editors, *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach*, volume 3562 of *LNCS*, pages 622–631. Springer, 2005.

[165] J. Sánchez, M. Galán, and E. Rubio. Applying a traffic lights evolutionary optimization technique to a real case: "Las Ramblas" area in Santa Cruz de Tenerife. *IEEE Transactions on Evolutionary Computation*, 12(1):25–40, 2008.

[166] J. Sánchez, M. Galán, and E. Rubio. Evolutionary computation applied to urban traffic optimization. In W. Kosinski, editor, *Advances in Evolutionary Algorithms*, pages 421–442. I-Tech Education and Publishing, 2008.

[167] H. Schepperle and K. Böhm. Auction-based traffic management: Towards effective concurrent utilization of road intersections. In *IEEE Joint Conference on E-Commerce Technology (CEC08) and Enterprise Computing, E-Commerce and E-Services (EEE08)*, pages 105–112, 2008.

[168] H. Schmeck. Organic Computing – A new vision for distributed embedded systems. In *Proceedings of the 8th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05)*, pages 201–203, 2005.

[169] C. Schmidt. *Evolutionary computation in stochastic environments*. PhD thesis, Institut für Angewandte Informatik und Formale Beschreibungsverfahren (AIFB), Universität Karlsruhe (TH), 2007.

[170] J.-D. Schmöcker, S. Ahuja, and M. G. Bell. Multi-objective signal control of urban junctions – Framework and a London case study. *Transportation Research Part C: Emerging Technologies*, 16(4):454–470, 2008.

[171] W. Schnabel and D. Lohse. *Grundlagen der Straßenverkehrstechnik und der Verkehrsplanung*. Verlag für Bauwesen, 1997.

[172] D. Schrank and T. Lomax. The 2009 Urban Mobility Report. Technical report, Texas Transportation Institute, 2009.

[173] H.-P. Schwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*. Birkhäuser, 1977.

[174] J. Sha'Aban, A. Tomlinson, B. Heydecker, and L. Bull. Adaptive traffic control using evolutionary algorithms. In *Proceedings of the 9th Meeting of the EURO Working Group on Transportation*, pages 372–377, 2002.

[175] A. G. Sims and K. W. Dobinson. The Sydney Coordinated Adaptive Traffic (SCAT) System – Philosophy and Benefits. *IEEE Transactions on Vehicular Technology*, 29(2):130–137, 1980.

[176] S. Smith. *A learning system based on genetic adaptive algorithms*. PhD thesis, Department of Computer Science, University of Pittsburgh, 1980.

[177] S. Smith. Flexible learning of problem solving heuristics through adaptive search. In *Proceedings of the 8th International Joint Conference on Articial Intelligence*, pages 421–425, 1983.

[178] R. Sterritt. Autonomic Computing. *Innovations in Systems and Software Engineering*, 1(1):79–88, 2005.

[179] R. Sterritt, M. Parashar, H. Tianfield, and R. Unland. A concise introduction to autonomic computing. *Advanced Engineering Informatics*, 19:181–187, 2005.

[180] A. Stevanovic. Review of adaptive traffic control principles and deployments in larger cities. In *Proceedings of mobil.TUM 2009 – International Scientific Conference on Mobility and Transport (CD-ROM)*, 2009.

[181] A. Stevanovic, P. T. Martin, and J. Stevanovic. VISGAOST: VISSIM-based genetic algorithm optimization of signal timings. In *Transportation Research Record 2035*, pages 59–68. Transportation Research Board, 2007.

[182] J. Stevanovic, A. Stevanovic, P. T. Martin, and T. Bauer. Stochastic optimization of traffic control and transit priority settings in VISSIM. *Transportation Research Part C: Emerging Technologies*, 16(3):332–349, 2008.

[183] C. Stone and L. Bull. For real! XCS with continuous-valued inputs. *Evolutionary Computation*, 11(3):299–336, 2003.

[184] B. Stroustrup. *The C++ Programming Language*. Addison Wesley, 3rd edition, 1997.

[185] D. Sun, R. F. Benekohal, and S. T. Waller. Multi-objective traffic signal timing optimization using non-dominated sorting genetic algorithm. In *Proceedings of the IEE Intelligent Vehicles Symposium*, pages 198–203, 2003.

[186] R. S. Sutton and A. G. Barto. *Reinforcement learning – An introduction*. MIT Press, 1998.

[187] A. S. Tanenbaum. *Computer Networks*. Pearson Education, 4th edition, 2002.

[188] D. Teodorović. Swarm intelligence systems for transportation engineering: Principles and applications. *Transportation Research Part C: Emerging Technologies*, 16(6):651–667, 2008.

[189] S. Tomforde, E. Çakar, and J. Hähner. Dynamic control of network protocols – A new vision for future self-organising networks. In *Proceedings of the 6th International Conference on Informatics in*

Control, Automation and Robotics – Intelligent Control Systems and Optimization (ICINCO-ICSO), pages 285–290, 2009.

[190] S. Tomforde, H. Prothmann, J. Branke, J. Hähner, M. Mnif, C. Müller-Schloer, U. Richter, and H. Schmeck. Observation and control of organic systems. In C. Müller-Schloer, H. Schmeck, and T. Ungerer, editors, *Organic Computing – A Paradigm Shift for Complex Systems*, Autonomic Systems, chapter 4.1, pages 325–338. Birkhäuser, 2011.

[191] S. Tomforde, H. Prothmann, J. Branke, J. Hähner, C. Müller-Schloer, and H. Schmeck. Possibilities and limitations of decentralised traffic control systems. In *WCCI 2010 IEEE World Congress on Computational Intelligence*, pages 3298–3306. IEEE, 2010.

[192] S. Tomforde, H. Prothmann, F. Rochner, J. Branke, J. Hähner, C. Müller-Schloer, and H. Schmeck. Decentralised progressive signal systems for organic traffic control. In S. Brueckner, P. Robertson, and U. Bellur, editors, *Proceedings of the 2nd IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2008)*, pages 413–422. IEEE, 2008.

[193] S. Tomforde, M. Steffen, J. Hähner, and C. Müller-Schloer. Towards an organic network control system. In J. G. Nieto, W. Reif, G. Wang, and J. Indulska, editors, *Autonomic and Trusted Computing – 6th International Conference (ATC 2009)*, volume 5586 of *LNCS*, pages 2–16. Springer, 2009.

[194] Transportation Research Board. Artificial Intelligence in Transportation. Transportation Research Circular E-C113, 2007.

[195] M. Treiber and A. Kesting. *Verkehrsdynamik und -simulation – Daten, Modelle und Anwendungen der Verkehrsflussdynamik*. Springer, 2010.

[196] TSS – Transport Simulation Systems. *AIMSUN 5.1 Microsimulator User's Manual*, 2008.

[197] UK Department for Transport. New car fuel consumption: The official figures, 1994.

[198] UK Department for Transport – Vehicle Certification Agency. New car fuel consumption & emission figures, 2009.

[199] United Nations Department of Economic and Social Affairs. World Urbanization Prospects – The 2007 Revision, 2008.

[200] M. Vasirani and S. Ossowski. A market-inspired approach to reservation-based urban road traffic management. In C. Sierra, C. Castelfranchi, K. S. Decker, and J. S. Sichman, editors, *Proceedings of the 8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '09)*, pages 617–624, 2009.

[201] VDE/ITG/GI. Positionspapier Organic Computing, 2003.

[202] E. I. Vlahogianni, J. C. Golias, and M. G. Karlaftis. Short-term traffic forecasting: Overview of objectives and methods. *Transport Reviews*, 24(5):533—557, 2004.

[203] A. Vogel. *Ein Ansatz zur Optimierung des Straßenverkehrs auf Knotenebene*. Dissertation, Institut für Neuroinformatik, Ruhr-Universität Bochum, 2001.

[204] A. Vogel, C. Goerick, and W. von Seelen. Evolutionary algorithms for optimizing traffic signal operation. In *Proceedings of the European Symposium on Intelligent Techniques (ESIT)*, pages 83–91, 2000.

[205] C. Volhard. Vorhersage der Verkehrsentwicklung für autonome Lichtsignal-Anlagensteuerungen. Masterarbeit, Institut für Systems Engineering – System und Rechnerarchitektur, Leibniz Universität Hannover, 2009.

[206] P. Vortisch, S. Menneni, and C. Sun. Kalibrierung von Fahrverhaltensparametern in der mikroskopischen Verkehrsflusssimulation mit Hilfe evolutionärer Algorithmen. *Straßenverkehrstechnik*, 52(5):274–279, 2008.

[207] J. Wahle, O. Annen, C. Schuster, L. Neubert, and M. Schreckenberg. A dynamic route guidance system based on real traffic data. *European Journal of Operational Research*, 131(2):302–308, 2001.

[208] F. V. Webster. *Traffic Signal Settings.* Road Research Technical Paper No. 39. UK Road Research Laboratory, Department of Scientific and Industrial Research, 1958.

[209] K. Weicker. *Evolutionäre Algorithmen.* Teubner, 2nd edition, 2007.

[210] R. Wiedemann. *Simulation des Straßenverkehrsflusses.* Habilitationsschrift, Institut für Verkehrswesen, Universität Karlsruhe (TH), 1974.

[211] C. J. Wilson, G. Millar, and R. Tudge. Microsimulation evaluation of benefits of SCATS-coordinated traffic control signals. In *Proceedings of the 85th Transportation Research Board Annual Meeting.* Transportation Research Board, 2006.

[212] S. W. Wilson. ZCS: A zeroth level classifier system. *Evolutionary Computation*, 2(1):1–18, 1994.

[213] S. W. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.

[214] S. W. Wilson. Generalization in the XCS classifier system. In J. R. Koza et al., editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 665–674. Morgan Kaufmann, 1998.

[215] S. W. Wilson. Get real! XCS with continuous-valued inputs. In P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *Learning Classifier Systems – From Foundations to Applications*, volume 1813 of *LNAI*, pages 209–219. Springer, 2000.

[216] S. W. Wilson. Mining oblique data with XCS. In P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *Advances in Learning Classifier Systems – 3rd International Workshop, IWLCS 2000*, volume 1996 of *LNAI*, pages 283–290. Springer, 2001.

[217] S. W. Wilson. Classifiers that approximate functions. *Natural Computing*, 1(2–3):211–233, 2002.

[218] S. W. Wilson. Three architectures for continuous action. In T. Kovacs, X. Llorà, K. Takadama, P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *Learning Classifier Systems – International Workshops, IWLCS 2003-2005*, volume 4399 of *LNAI*, pages 239–257. Springer, 2005.

[219] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2nd edition, 2005.

[220] L. A. Wolsey. *Integer Programming*. Wiley, 1998.

[221] M. Wünsche, S. Mostaghim, H. Schmeck, T. Kautzmann, and M. Geimer. Organic computing in off-highway machines. In *Second International Workshop on Self-Organizing Architectures (SOAR '10)*, pages 51–58. ACM, 2010.

[222] W. Zhizhou, S. Jian, and Y. Xiaoguang. Calibration of VISSIM for Shanghai expressway using genetic algorithm. In M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, editors, *Proceedings of the 2005 Winter Simulation Conference*, pages 2645–2648, 2005.

[223] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In K. Giannakoglou, D. Tsahalis, J. Periaux, K. Papailiou, and T. Fogarty, editors, *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems*, pages 95–100. CIMNE, 2002.

ORGANIC TRAFFIC CONTROL

Modern cities cannot be imagined without traffic lights controlling their road network. To handle the network's traffic efficiently, the traffic lights need to adapt their signalisation in response to changing demands. This requires shifting the signal plan specification from the design time to the run-time of the signal system.

This work builds on the generic observer / controller architecture proposed for Organic Computing to facilitate the shift. A two-levelled learning mechanism optimises an intersection's signal plan on-line while a distributed coordination mechanism establishes progressive signal systems (or "green waves") in response to the current traffic demand. Thereby, delays and stops in the road network can be reduced such that fuel consumption rates and pollution emissions are lowered. The two-levelled learning mechanism is generally applicable to a wide area of safety- and performance-critical applications that have been inept for on-line optimisation before.

ISBN 978-3-86644-725-7